305-CD-010-001

EOSDIS Core System Project

Release A SDPS Planning Subsystem Design Specification for the ECS Project

July 1995

Hughes Information Technology Corporation
Landover, MD

Release A SDPS Planning Subsystem Design Specification for the ECS Project

July 1995

Prepared Under Contract NAS5-60000
CDRL Item #046

SUBMITTED BY

Parag N. Ambardekar /s/	7/27/95
Parag Ambardekar, Release A CCB Chairman EOSDIS Core System Project	Date

Hughes Information Technology Corporation
Landover, Maryland

This page intentionally left blank.

Preface

This document is one of sixteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release A of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-004, which provides an overview of the subsystems and 305-CD-018, the Data Dictionary, for those reviewing the object models in detail. A Release A SDPS and CSMS CDR Review Guide (510-TP-002) is also available.

The SDPS and CSMS subsystem design specification documents for Release A of the ECS Project include:

305-CD-004	Release A Overview of the SDPS and CSMS Segment System Design Specification
305-CD-005	Release A SDPS Client Subsystem Design Specification
305-CD-006	Release A SDPS Interoperability Subsystem Design Specification
305-CD-007	Release A SDPS Data Management Subsystem Design Specification
305-CD-008	Release A SDPS Data Server Subsystem Design Specification
305-CD-009	Release A SDPS Ingest Subsystem Design Specification
305-CD-010	Release A SDPS Planning Subsystem Design Specification
305-CD-011	Release A SDPS Data Processing Subsystem Design Specification
305-CD-012	Release A CSMS Segment Communications Subsystem DesignSpecification
305-CD-013	Release A CSMS Segment Systems Management Subsystem Design Specification
305-CD-014	Release A GSFC Distributed Active Archive Center Implementation
305-CD-015	Release A LaRC Distributed Active Archive Center Implementation
305-CD-016	Release A MSFC Distributed Active Archive Center Implementation
305-CD-017	Release A EROS Data Center Distributed Active Archive Center Implementation
305-CD-018	Release A Data Dictionary for Subsystem Design Specification
305-CD-019	Release A System Monitoring and Coordination Center Implementation

Object models presented in this document have been exported directly from CASE tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) at URL http://edhs1.gsfc.nasa.gov.

This document is a contract deliverable with an approval code 2. As such, it does not require formal Government approval, however, the Government reserves the right to request changes within 45 days of the initial submittal. Once approved, contractor changes to this document are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Corporation
1616 McCormick Drive
Landover, MD 20785

Abstract

This document describes the Release A CDR design for the SDPS Planning subsystem. It defines the Planning Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements. The subsystem comprises of two CIs: the PLANG CSCI and the PLNHW HWCI.

Keywords:

Planning, SDPS, Release A, OMT, PDPS

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Final
iii through xvi	Final
1-1 through 1-2	Final
2-1 through 2-2	Final
3-1 through 3-12	Final
4-1 through 4-	Final
5-1 through 5-10	Final
A-1 through A-14	Final
AB-1 through AB-2	Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
305-CD-010-001	Final	July 1995	95-0473

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

2. Related Documentation

3. Subsystem Overview

4. PLANG - Production Planning CSCI

5. Planning Subsystem Hardware CI

Figures

Tables

Appendix A.

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

This Release A SDPS Planning Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) Item 046, with requirements specified in Data Item Description (DID) 305/DV10, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management segment (CSMS) and the Science and Data Processing Subsystem (SDPS) for Release A.

1.2 Purpose and Scope

The Release A SDPS Planning Subsystem Design Specification defines the progress of the design. It defines the Planning Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements.

This document reflects the June 21, 1995 Technical Baseline maintained by the contractor configuration control board in accordance with ECS Technical Direction No.11, dated December 6, 1994.

1.3 Status and Schedule

This submittal of DID 305/DV10 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. A previous version of this submittal was reviewed during the SDPS Preliminary Design Review (PDR) and reflects changes to the design which resulted from that review. The PDR also triggered a number of follow up actions in response to Review Item Discrepancies (RID) the results of which have been incorporated into this Critical Design Review (CDR) version of this document.

1.4 Organization

The document is organized to describe the Planning subsystem design as follows:

Section 1 provides information regarding the identification, scope, status, and organization of this document.

Section 2 provides a listing of the related documents, which were used as source information for this document.

Section 3 provides an overview of the subsystem, focusing on the high-level design concept. This provides general background information to put the Planning subsystem into context.

Sections 4 contain the structure of the computer software configuration item (CSCI) comprising the Planning subsystem.

Section 5 contains the hardware configuration item (HWCI) design of the Planning subsystem.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this volume.

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which the scope and content of this Planning Subsystem Design Specification is derived.

194-207-SE1-001	System Design Specification for the ECS Project

2.2 Applicable Documents

The following documents are referenced within this Planning Subsystem Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

209-CD-001-001	Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet
209-CD-002-001	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System
209-CD-003-001	Interface Control Document Between EOSDIS Core System (ECS) and EOS-AM Project for AM-1 Spacecraft Analysis Software
209-CD-004-001	Data Format Control Document for the Earth Observing System (EOS) AM-1 Project Data Base
209-CD-005-002	Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF)
209-CD-006-002	Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC)
209-CD-007-002	Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS)
209-CD-008-002	Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC)
209-CD-009-002	Interface Control Document Between EOSDIS Core System (ECS) and the Marshall Space Flight Center (MSFC) Distributed Active Archive Center (DAAC)
209-CD-011-002	Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System
305-CD-002-002	Science Data Processing Segment (SDPS) Segment Design Specification for the ECS Project
305-CD-003-002	Communications and System Management Segment (CSMS) Design Specification for the ECS Project

308-CD-001-004	Software Development Plan for the ECS Project
313-CD-004-001	Release A CSMS/SDPS Internal Interface Control Document for the ECS Project
423-41-03	Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document

3. Subsystem Overview

The Planning subsystem is responsible for supporting operation staff in managing the data production activities at a site. The Planning subsystem assists the operations staff in performing two major functions:

- defining the data processing tasks to be performed at a site; and
- generating efficient plans for the scheduling of those tasks.

In addition the Planning subsystem is responsible for coordinating the production with the Data Server and Data Processing subsystems to achieve a highly automated production system.

3.1 Context

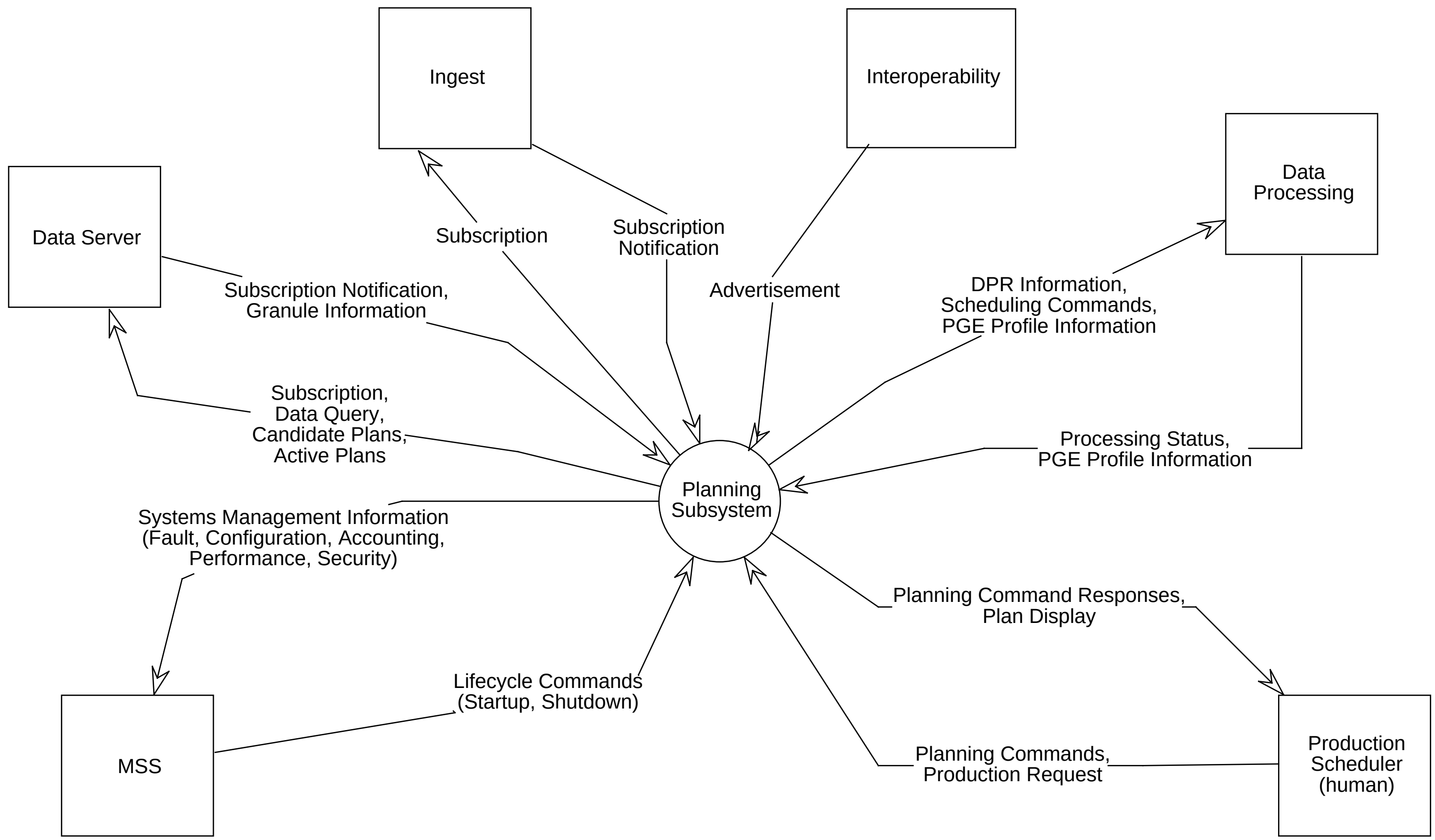
A context diagram illustrating the relationships between the Planning subsystem and other subsystems is shown in Figure 3.1-1, and summarized in Table 3.1-1.

The context diagram shows only those interfaces which are a part of the Tropical Rainfall Measuring Mission (TRMM) release Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS). Interfaces that provide additional capabilities will be described in the detailed design document at the appropriate future release. Most notably the interface for on-demand requests from the Science Data Server will be developed in the AM-1 release.

The Planning subsystem interfaces with the Algorithm Integration and Test Tools Computer Software Configuration Item (CSCI) within the Data Processing subsystem for information on Product Generation Executives (PGE). The information that is collected is referred to as a PGE Profile. This includes information on the PGE executable, the input data type(s) it requires, the output data type(s) it generates, the PGE resource requirements - hardware platform, memory, disk storage etc. The Planning subsystem provides storage for the profiles to permit the Algorithm Integration and Test Tools to insert, retrieve and modify PGE information.

The primary interface of the Planning subsystem with the Data Processing subsystem is to describe the PGEs that need to be run in order to fulfill the production goals. A Data Processing Request (DPR) describes a run of a PGE to the Data Processing subsystem. A DPR describes the specific input granules, output filenames, and run-time parameters for a PGE, as well as dependencies and target run-times. The Data Processing subsystem provides status and processing completion information to the Planning subsystem.

The Planning User Interface provides a means of human interaction with the Planning Subsystem. Through this interface a user can enter Production Requests. A Production Request describes an order for data that is to be produced by the Data Processing subsystem. Production Requests may signify the need for processing of new data (Standard Production Requests, a.k.a. standing orders) or the need for the reprocessing of data (Reprocessing Production Requests). The planning subsystem takes these Production Requests and uses the PGE profile information to work out the Data Processing Requests that will be required to fulfill the Production Request. The Planning User Interface is also used to issue commands to initiate plan creation, plan activation and the cancelling of a plan, as well as providing reports / status of progress within a plan.



Planning Commands
Create Plan Command, Activate Plan Command, Cancel Plan Command,
Update Plan Command

Scheduling Commands
Schedule Job Command, Cancel Job Command, Update Job Command,
Release Job Command

Planning Command Response
Production Request Response, Create Plan Command Response,
Active Plan Command Response, Cancel Plan Command Response

Figure 3.1-1. Context Diagram

The Planning subsystem queries the Data Server holdings for existence of data required in some reprocessing. The Data Server responds with granule information (identification, metadata, location). The Data Server also provides the subscription services needed by Planning to determine when new data are available for processing. The Planning subsystem generates the subscriptions to be entered into the Data Server. The Data Server sends a notification when data that fulfill the subscription are inserted into the Data Server.

The Ingest subsystem also provides a subscription service (via the Ingest Data Server) for notification of the arrival of L0 data from SDPF. The Advertising CSCI within the subsystem provides the advertisement data that is required by the Planning subsystem in order to generate a subscription.

The Planning subsystem uses the Document Data Server within the Data Server subsystem to store production plans. The Document Data Server then make those plans available to the user community.

The Planning Subsystem also has an interface to the Management Subsystem (MSS). The Planning subsystem is responsible for sending MSS fault management data, accounting data, security data, and performance data.

Table 3.1-1. Planning Subsystem Interfaces (1 of 2)

Source	Destination	Data Types	Data Volume	Frequency
Production Scheduler	Planning	Production request	low	as required
Production Scheduler	Planning	create plan command	low	as required
Production Scheduler	Planning	activate plan command	low	as required
Production Scheduler	Planning	update plan command	low	as required
Production Scheduler	Planning	cancel plan command	low	as required
Planning	Production Scheduler	Production request response	low	in response to standard production request
Planning	Production Scheduler	create plan command response	low	in response to create plan command
Planning	Production Scheduler	activate plan command response	low	in response to activate plan command
Planning	Production Scheduler	cancel plan command response	low	in response to cancel plan command
Planning	Production Scheduler	plan (display)	medium	as required
Planning	Data Processing	data processing request information	low	in response to activating a plan
Planning	Data Processing	schedule job command	low	in response to activating plan

Table 3.1-1. Planning Subsystem Interfaces (2 of 2)

Source	Destination	Data Types	Data Volume	Frequency
Planning	Data Processing	cancel job command	low	in response to activating plan
Planning	Data Processing	release job command	low	in response to activating plan
Planning	Data Processing	update job command	low	in response to activating plan
Data Processing	Planning	Processing status	low	in response to an update job command
Data Processing (AI&T)	Planning	PGE profile information	low	as required
Planning	Data Processing (AI&T)	PGE profile information	low	as required
Planning	Data Server (Document Data Server)	candidate plan, active plan	medium	as required for archiving
Planning	Data Server	data query	low	as required
Planning	Data Server	subscription	low	as required
Data Server	Planning	subscription notification	low	when input data are available
Data Server	Planning	granule information	low	in response to query
Planning	Ingest	subscription	low	as required
Ingest	Planning	subscription notification	low	one/few times per day as L0 data arrives from SDPF
Planning	MSS	fault management	low	as required
Planning	MSS	configuration management	low	as required
Planning	MSS	accounting data	low	as required
Planning	MSS	security data	low	as required
Planning	MSS	performance data	low	as required
MSS	Planning	lifecycle commands	low	as required

In the table, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column . The frequency information will be updated as the interfaces are fully defined.

3.2 Subsystem Overview

This section describes the key drivers for the Planning subsystem and how the design responds to them. The section starts with a discussion of the use of Commercial-off-the-shelf (COTS) within Planning and Data Processing and describes the key changes to the Planning subsystem since the preliminary design specification, brought about by the selection of COTS.

3.2.1 Use of COTS within the Planning and Data Processing Subsystems

Between Preliminary Design Review (PDR) and Critical Design Review (CDR) the Planning and Data Processing subsystems have focused much attention on the evaluation and selection of COTS. Managing systems involving large numbers of processing tasks is not a problem unique to ECS; robust vendor provided software exists to facilitate a high degree of automation in these systems. When carefully selected, COTS significantly reduces both the schedule and technical risks and can provide capabilities that would otherwise take considerable effort to develop. The evaluation and selection process is described in the Data Processing Subsystem Design Specification. It is important to emphasize that the selection of COTS that provides maximum benefits to ECS has resulted in a change to the PDR architecture and design of both the Planning and Data Processing subsystems. This will be outlined in Section 3.2.2.

3.2.1.1 Brief Description of Job Scheduling COTS

The main focus of COTS evaluation within the Planning and Data Processing subsystems has been to select a Job Scheduling Engine to provide the automated production management capabilities required. The COTS that have been selected is a combination of packages called AutoSys and AutoXpert.

AutoSys provides robust capabilities for managing schedules of Jobs. Jobs are defined within the AutoSys database and are initiated on remote machines when the dependencies of a Job have been met, and the resources required to complete the Job are available. AutoSys basically comprises of three components:

- AutoSys RDBMS - contains the job definition and dependency information;
- Event Processor - reads DBMS to determine actions to be taken; and
- AutoSys Remot Agent - initiates a job on a remote process.

The basic production sequence of AutoSys is shown diagrammatically in Figure 3.2-1.

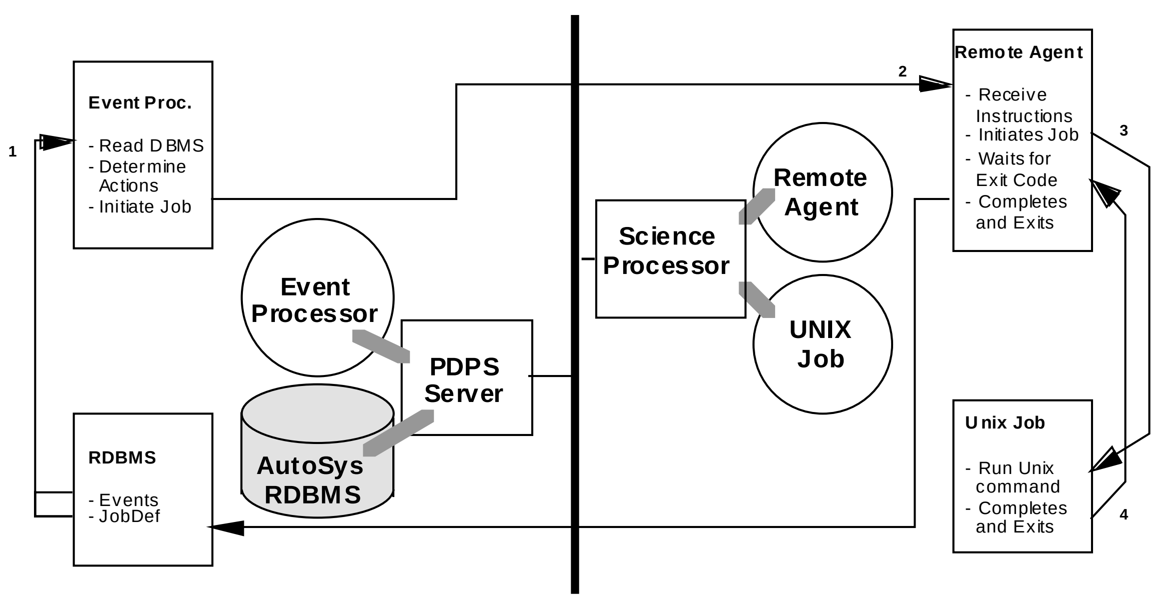


Figure 3.2-1. Basic Capability of AutoSys

AutoXpert provides advanced capabilities for monitoring the schedule of Jobs and predicting their completion time. The graphic display provided by the AutoXpert provides for three views onto the production schedule:

- Timeline or GANTT style view, showing the current status according to schedule and estimated completion times for tasks;
- Job Network or PERT style view, showing the current status according to the task dependencies; and,
- Resource style view, showing the current status according to allocation of tasks to machines.

AutoXpert also provides capabilities to perform "what-if" scenarios on the schedule to predict the impact of failures within the job network on the downstream dependent tasks.

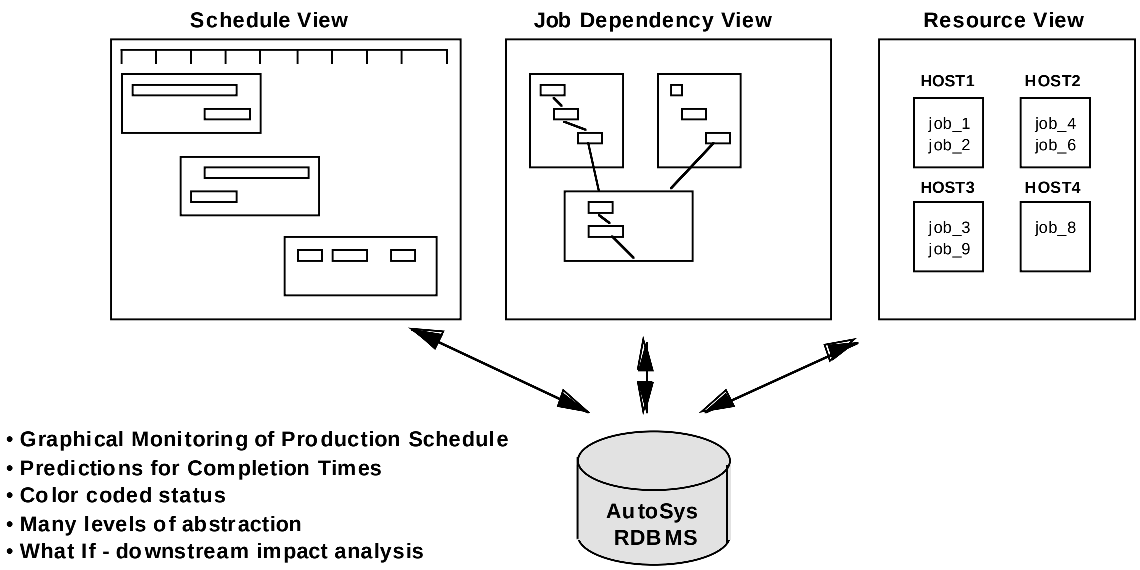


Figure 3.2-2. Graphical Schedule displays provided by AutoXpert

3.2.2 Summary of Changes to the Planning Subsystem

As described in the COTS evaluation paper, the goal of the selection was to obtain the packages which provide the maximum of the capabilities required by the ECS. No COTS solution exists to cover the entire ECS Planning and Data Processing subsystems. The design presented in the rest of this document describes those areas which are ECS specific or areas where the requirements can not be covered by COTS. In the area where the requirements can not currently be covered by COTS it should be emphasized that the ECS is also actively pursuing the vendors to improve their capabilities.

The basic capabilities of the Planning subsystem described in the preliminary design have not changed. Rather, these capabilities have been enhanced by the AutoSys/AutoXpert selection. What has changed is the division of responsibilities within the Planning and Data Processing subsystems. A major goal in re-architecting the system after the selection was to ensure that the interfaces to

COTS are within one subsystem. This ensures that interfaces to the COTS can be appropriately encapsulated to give later flexibility augmenting, modifying or replacing the underlying COTS as ECS matures.

The Job Scheduling COTS is appropriately made a component of the Data Processing subsystem, which accounts for the reallocation of some of the Planning capabilities. The key changes to the Planning design since PDR are within the Production Management capabilities. The two main activities performed here were:

1) to coordinate the production by providing a Data Processing Request to the Data Processing subsystem when all the data required for the task are present at the Data Server,
2) to provide a display of the active production, and it's status according to the plan.

These two capabilities have been split into separate CSCs. The (new) subscription manager CSC, which is part of this subsystem, provides the first of these capabilities. The graphic capabilities of AutoXpert in the Data Processing subsystem provides for the second.

Other changes that have been brought on due to the selection of AutoSys and AutoXpert pertain to the sequence of events "activating" a plan. This now involves rolling a portion of the "long term" plans generated in the Planning subsystem into AutoSys. The design and scenarios that describe this procedure are presented in Section 4 of this document.

3.2.3 Key Design Drivers

As outlined in the overview the planning subsystem has three main responsibilities:

- Defining the goals for data production,
- Preparing plans for production, and
- Coordinating the production from those plans.

Each of these areas are discussed in turn in order to describe how the subsystem responds to the key drivers in these areas.

3.2.3.1 Defining Production Goals

3.2.3.1.1 PGE Profiles

The Planning subsystem stores a description of the PGE in order to be able to determine the correct PGEs and associated input required to generate a product. Each PGE has individual data requirements. These data requirements are captured within the PGE Profile to allow the Planning subsystem to select the appropriate data for a PGE either from the Data Server (in the case of reprocessing) or from its own predictions of data availability (in the case of standard production). The PGE profile is a key component of the Planning subsystem that allows the Planning subsystem to determine which PGEs have to be run to create a given data set (the term Data Processing Requests is used within the ECS to describe an individual run of a PGE).

3.2.3.1.2 Activation Rules

There is a fair degree of diversity in the scheduling characteristics of PGEs that are being developed for ECS. The most standard category of PGE describes those that require a well defined period of data as input and produce one or more output data products. For example, the CERES subsystem 1 geolocation and earth radiance calibration PGE requires one days worth of L0 data and ephemeris data and produces a single product file of bidirectional scan data, as well as 24 files of instrument earth scans, each containing 1 hour of data. However, there are many other categories with more complex scheduling requirements (PGEs scheduled per orbit, PGEs that mosaic data into geographical tiles, PGEs with alternative data sets that are determined at run time). These types of PGE have been characterized for the ECS through the work of the Ad Hoc Working Group on Production (AHWGP) and are described in terms of activation rules.

Not all of the activation rules will be supported within the Planning subsystem at the TRMM Release of ECS. The maturity of the PGE definitions as well as schedule constraints make it suitable to develop the capabilities incrementally. Fortunately object oriented methods permit management of this in a well defined manner; the different PGE types are modeled within a “specialization” hierarchy. Each modeled PGE class containing its own methods determining when it will be scheduled. This allows the Planning subsystem to develop the key abstractions required to support the CERES and LIS processing algorithms, and to add further capabilities at later releases.

3.2.3.1.3 Production Requests

The ECS supports both routine production (at this release) and ad-hoc or on-demand production (at next release). The goal of ECS is to produce science data sets to facilitate earth science research. Consistently generated data spanning long time frames are required for much of this research; therefore, many of the tasks performed within the data processing subsystem will be highly routine with the same task being performed (albeit on different data) a large number of times per month. As described above, the formulation of a Data Processing Request is a non-trivial task, and therefore a high degree of automation is required to aid in this process. The Production Request is a key abstraction which describes an “order” for the generation of a data set over a period of time. The Production Request is used along with the PGE profile to generate the Data Processing Requests; typically one production request will generate many data processing requests. The Planning subsystem will determine a default PGE to generate this product; however, this predetermination may be overridden within the Production Request. Similarly, user parameters may be provided to override their defaults defined within the PGE profile. In addition, the Planning subsystem provides the capabilities to review and modify the individual Data Processing Requests defined from the Production Request to ensure that the correct task is defined to achieve the production goal. Not all Production Requests have to be put into production, Production Requests can be defined to describe backup options for processing. On identification of a problem within the production chain, the tasks defined for a backup Production Request may be quickly scheduled.

3.2.3.2 Preparing Plans for Production

3.2.3.2.1 Planning

The ability to generate production plans is important to both operations and users for the following reasons:

- *Staffing Projections:* A long term projection describing the data products that will be generated at a site allows organizations to manage staffing to support the anticipated production, for example, to cover on-site or off-site QA requirements.

- *Disconnects:* Dependencies on data not planned for production can be identified, which then allows corrective steps to be taken.
- *Efficiency:* Proper organization of the data production activities within a plan ensures that the production will utilize resources in an efficient manner.

There is no explicit planning horizon that is hard coded into the Planning subsystem software. The system is sized to support generating a number of candidate plans on the order of a month in advance.

3.2.3.2.2 Publishing Plans

The Planning subsystem provides the capability to publish plans that are generated at a site. The Document Data Server is the most appropriate repository for these plans, since the plans can then be made available to WWW access. The published plans will also be stored within a WAIS database so that, using a simple HTML forms interface, a user may select the plans for a specific period. A small set of metadata will describe the plans, to allow the user to select a plan based on period, data product types which are scheduled in the plan, as well as operator comments. A number of plan formats will be stored within the Document Data Server including a simple ASCII report as well as a more electronically parsable format. The ability to include meaningful graphics within another possible format will be investigated.

In future releases the ability to subscribe to plans will be supported so that users may be automatically alerted to delays in production. Until that point a simple script will be provided to inform a prepared list of recipients to changes of Plans.

3.2.3.3 Managing the Production

3.2.3.3.1 Schedule/Data Driven Production

The plans generated in the Planning subsystem describe a prediction of the activities that will occur in the Data Processing subsystem. This prediction is only a forecast of the activities that will occur and a recipe for the production under nominal conditions. It is important to emphasize that the production isn't driven directly from the plan. As an example, if a PGE is predicted to start at 3.00 in the afternoon it doesn't mean that the PGE will be executed in the Data Processing subsystem at 3.00 precisely. The Planning subsystem always ensures that the data are present within the system before an activity is released to run. The Data Processing subsystem ensures that all of the required resources are available to permit that activity to complete. Jobs are queued within the Data Processing subsystem so that (within the constraints of priority and job dependencies) the resources will be kept efficiently occupied.

3.2.3.3.2 Using the Plan

There are many events that can impact a plan: delays in data arrival, poor performance of system components, etc. There is no need to replan in circumstances like these. The Planning and Data Processing architecture and design are robust to many fault situations. A replan is only really required when the production goals need to be redefined to accommodate an unanticipated circumstance. For example, a repeated failure within the science data processing might indicate that it would be appropriate to switch from standard production to concentrate on some reprocessing objectives.

As previously stated, plans generated in the Planning subsystem describe a prediction of the activities that will occur in the Data Processing subsystem. The system is tolerant to changes in schedule, but these changes usually indicate a circumstance that needs to be brought to the attention of the operators. The forecasts provided by the Planning subsystem are used within the Data Processing subsystem to define alarms to alert the operator of deviations against the schedule. Delays in the arrival of data, which may not provoke any exception in the system that would be identified as an error, will thus still be noticed and flagged.

3.2.3.3.3 Bypassing the Plan

It is understood that, especially during early mission activities, it is frequently desirable to be able to schedule jobs individually. Operators require the capability to insert a job into the stream without necessarily planning this task. The net result of inserting a job into the production stream in an ad-hoc manner is that the forecast times for completion as determined by Planning may no longer be accurate. However, the benefits that this capability bring are clear. The planning subsystem will permit authorized operations staff to enter a Production Request and schedule the resulting Data Processing Requests to the Data Processing subsystem.

3.2.3.4 Distribution of Planning Capabilities

3.2.3.4.1 SCF access to Planning Capabilities

It is acknowledged that, the definition of Production Requests - specifying the data processing goals - is tightly coupled to science objectives of calibration and validation of instruments and algorithms, especially during early mission life. Since the responsibility of production ultimately resides with the Distributed Active Archive Center (DAAC), there is no intent to distribute the Planning subsystem capabilities away from the DAAC. However, to encourage the collaboration between the different teams with stakes in the production of science data, the Planning subsystem capabilities can be made available to remote access. Kerberized remote login procedures for special users and limited access permissions to the Planning subsystem database tools will ensure that DAACs can maintain secure authority over use of these capabilities.

3.2.4 Performance

The Planning subsystem design has been prepared to meet the overall performance objectives of the TRMM Release time period. This is accomplished through the design of both the software and hardware. An estimate of the daily average of TRMM-related processing jobs to be planned, scheduled has been extracted from the Technical Baseline and presented in the table below.

Table 3.2-1. TRMM Release Planning Workload

TRMM Processing PGEs	Processing Location	Activations per Day
CERES (12 SubSystems)	LaRC	61.02
LIS	MSFC	14.56

An assessment of the performance associated with this workload level has been performed which indicates that a relatively small server could adequately support this activity at both locations. The final design will reflect the ECS AM-1 Release (B) workload level which is more challenging for the Planning Subsystem, but certainly within the range of available technology. This final design will be performed after the AutoSys/AutoXpert COTS package has been installed at the EDF with the associated Sybase database server and benchmarked with typical workloads to insure adequate performance for operations. It is felt that this is the most prudent approach considering the importance associated with the planning COTS within the Planning subsystem. It is also estimated that the raw data storage requirements for the TRMM Release for the purposes of planning are of the order of 2.2 Mbytes at LaRC and 0.5 Mbytes at MSFC. The design for these locations will also be sized to accommodate the ECS AM-1 Release rather than the TRMM Release capacities.

In terms of the more hardware specific RMA requirements, the design incorporates redundant servers for planning purposes with mirrored disks to contain the Planning database that supports both the Planning Subsystem and the Processing Subsystem. This is to insure the uninterrupted management of the planning and processing activity, and will insure that the mean time to restore service will be well within the 30 minute recovery time.

In addition, the design of the Planning and Data Processing subsystems has been prepared to minimize needless disk I/O by planning science data processing jobs to occur such that the data output from one PGE which is required for input by a second independent PGE is retained after the end of the first PGE and made available as input to the second PGE. In this way needless disk I/O and network loading is avoided, insuring that the systems performance goals can be met with the least cost.

3.2.5 Subsystem Structure

The Planning subsystem consists of two CIs: the Production Planning CSCI and the Production Planning HWCI. The Planning CSCI consists of software to meet the subsystem requirements. The Planning HWCI consists of the hardware resources to support the Planning software.

This page intentionally left blank.

4. PLANG - Production Planning CSCI

4.1 CSCI Overview

The Production Planning CSCI consists of a number of utilities and server applications as shown in Figure 4.1-1. The distinction between these two classes of programs is useful when describing the software design:

- Utilities are programs that perform well defined tasks and are invoked at the operator's requests. A utility is usually exited when the task is complete; the system does not depend on the utility being active at any given time. As an example, a mail editor is a common utility program, invoked whenever a user wishes to generate a mail message.
- Servers are programs that perform ongoing tasks and have to be active at all times for the subsystem to carry out its allocated functions. An example here is the mail daemon; the mail won't be delivered unless the daemon is running.

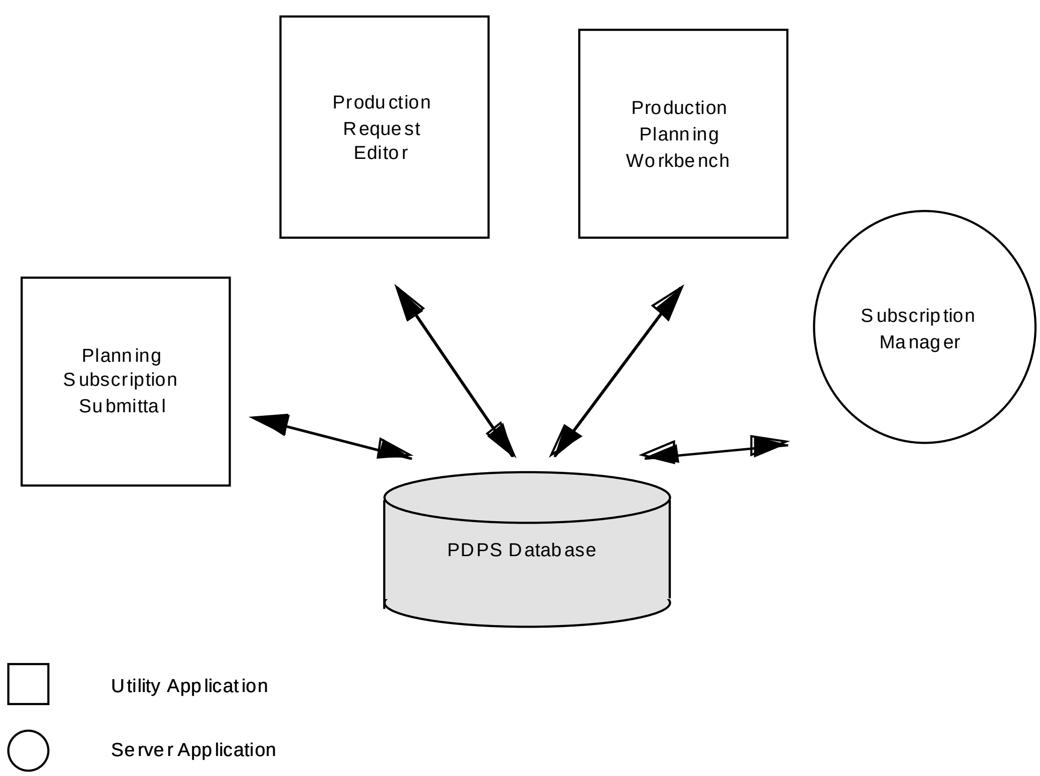


Figure 4.1-1. CSCI Overview

The allocation of the Planning subsystem capabilities to distinct applications accounts for the need to provide distinct interfaces for distinct activities, and to restrict access to planning functions to classes of users. The applications define an integrated set of tools with well-defined functions to supplement the COTS components. The allocation of capabilities to applications within the Planning CSCI also accounts for the need to permit independent development of the components as the capabilities of the ECS evolve. The purpose of each application is briefly outlined below.

4.1.1 PDPS Database

At the heart of the Planning CSCI is the Planning and Data Processing System (PDPS) Database (and the implied database server). This provides the persistent storage for data and facilitates the sharing of this data between the applications. The database also provides security, fault tolerance, and marshals request for concurrent access to data. The objects which are built from the persistent data within the PDPS database are indicated within the object design. The schema for the database has been driven from the object model and is described in the SDPS Database Design and Schema Specification CDRL.

4.1.2 Production Request Editor

This application allows the user to submit production requests that describe the data products to be produced. The application uses the PGE descriptions (profiles) entered during AI&T in order to work out the tasks - Data Processing Requests - that in sum meet the request. The application provides the capabilities to add, modify, and delete Production Requests, as well as review and modify the resulting Data Processing Requests. The production request editor is identified as a distinct application and separate from the workbench in anticipation that defining production requests will be a discrete activity, unrelated to the “planning” of these events.

The Production request editor may also be used by authorized operations staff to schedule Data Processing Requests derived from a production request directly to the Data Processing subsystem.

4.1.3 Production Planning Workbench

The application is used to prepare a schedule for the production at a site, and forecast the start and completion times of the activities within the schedule. These functions provided by the workbench include the following high-level activities:

- Candidate Plan Creation -- from the production requests prepared by the Production request Editor;
- Plan Activation -- activating a candidate plan;
- Updating the Active Plan -- feedback from the processing into the active plan; and
- Canceling/Modifying the Active Plan.

As described previously, activating a plan entails rolling a portion of a selected plan into the AutoSys COTS. This “schedule” is then managed within the Data Processing subsystem. The forecast times generated within the planner are used to set up operator alerts that would make the operator aware of gross departures from the predicted schedule. The production planning workbench can periodically update it's predictions using feedback from the AutoSys.

4.1.4 Planning Subscription Editor

This application provides the capabilities required to submit subscriptions to the Data Servers responsible for the storage of ingested data. Registration of a subscription at a Data Server is required for the Planning CSCI to receive notification when data arrive within the ECS. At this

Release, the submission of subscriptions will be managed as an operator initiated activity, although this may be automated at a later date.

4.1.5 Subscription Manager

The Subscription Manager is used to manage the receipt of a subscription notification from the Data Server. Subscription notification is used to notify Planning on the arrival of input data required by a given PGE. The Subscription Notification contains Universal References (URs) which are pointers to the data objects stored in the Data Server. The Subscription Manager updates the PDPS database to indicate when data become available. When all input data for a Data Processing Request is available, the job defined for that Data Processing Request is released within the Data Processing subsystem.

4.2 CSCI Context

Figure 4.2-1 and Table 4.2-2 illustrate the CSCI interfaces within the PLANG CSCI.

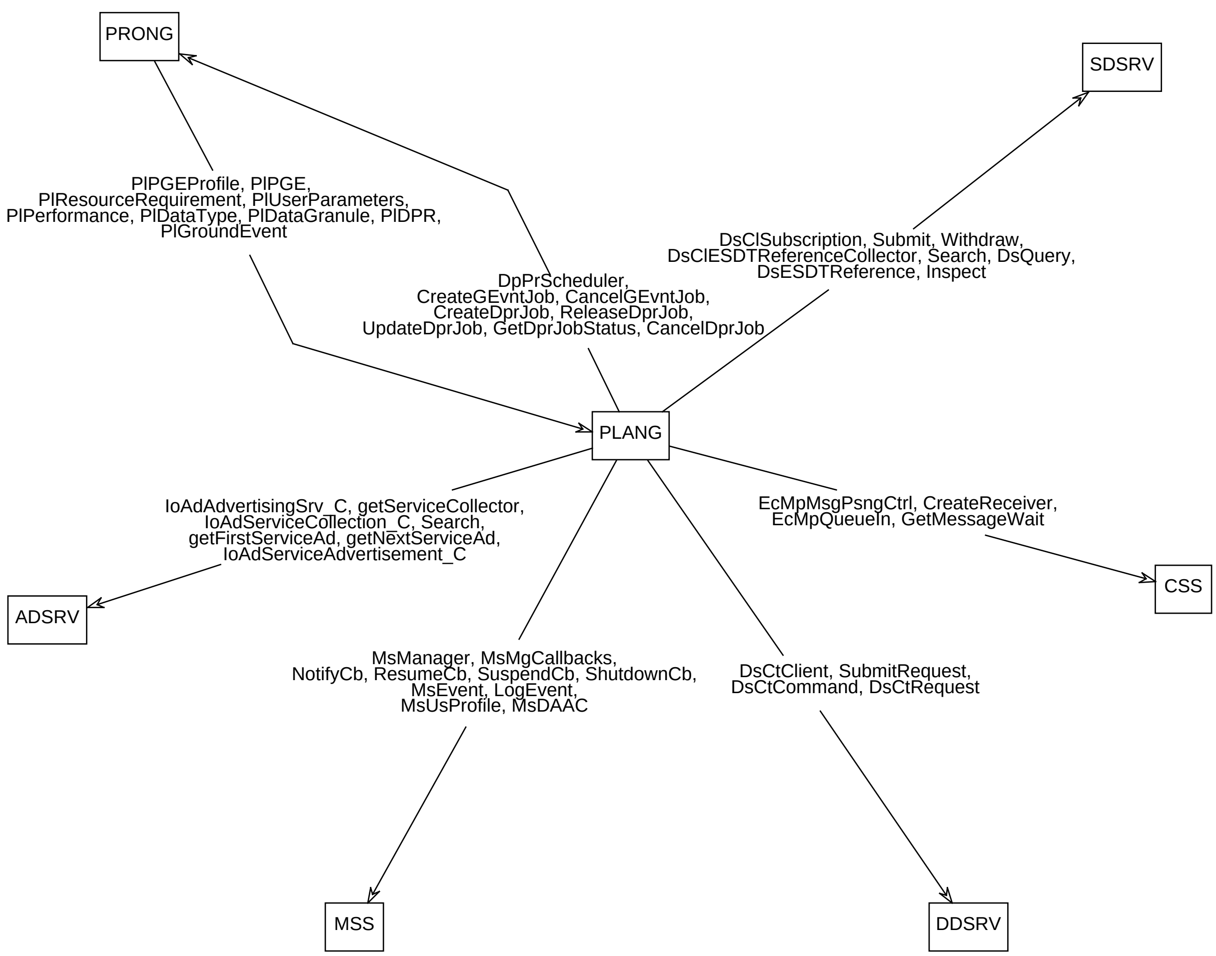


Figure 4.2-1. CSCI Event Flow Context Diagram

Table 4.2-2. CSCI Event Flow Summary (1 of 2)

Sender	Receiver	Event Name
PLANG	PRONG	CancelDprJob
PLANG	PRONG	CancelGEvntJob
PLANG	PRONG	CreateDprJob
PLANG	PRONG	CreateGEvntJob
PLANG	CSS	CreateReceiver
PLANG	PRONG	DpPrScheduler
PLANG	SDSRV	DsClESDTReferenceCollector
PLANG	SDSRV	DsClSubscription
PLANG	DDSRV	DsCtClient
PLANG	DDSRV	DsCtCommand
PLANG	DDSRV	DsCtRequest
PLANG	SDSRV	DsESDTReference
PLANG	SDSRV	DsQuery
PLANG	CSS	EcMpMsgPsngCtrl
PLANG	CSS	EcMpQueueIn
PLANG	PRONG	GetDprJobStatus
PLANG	CSS	GetMessageWait
PLANG	SDSRV	Inspect
PLANG	ADSRV	IoAdAdvertisingSrv_C
PLANG	ADSRV	IoAdServiceCollection_C
PLANG	ADSRV	IoAdServiceAdvertisement_C
PLANG	MSS	LogEvent
PLANG	MSS	MsDAAC
PLANG	MSS	MsEvent
PLANG	MSS	MsManager
PLANG	MSS	MsMgCallbacks
PLANG	MSS	MsUsProfile
PLANG	MSS	NotifyCb
PRONG	PLANG	PlDPR
PRONG	PLANG	PlDataGranule
PRONG	PLANG	PlDataType
PRONG	PLANG	PlGroundEvent
PRONG	PLANG	PlPGE
PRONG	PLANG	PlPGEProfile
PRONG	PLANG	PlPerformance
PRONG	PLANG	PlResourceRequirement
PRONG	PLANG	PlUserParameters
PLANG	PRONG	ReleaseDprJob
PLANG	MSS	ResumeCb
PLANG	SDSRV	Search
PLANG	ADSRV	Search

Table 4.2-2. CSCI Event Flow Summary (2 of 2)

Sender	Receiver	Event Name
PLANG	MSS	ShutdownCb
PLANG	SDSRV	Submit
PLANG	DDSRV	SubmitRequest
PLANG	MSS	SuspendCb
PLANG	PRONG	UpdateDprJob
PLANG	SDSRV	Withdraw
PLANG	ADSRV	getFirstServiceAd
PLANG	ADSRV	getNextServiceAd
PLANG	ADSRV	getServiceCollector

4.3 CSCI Object Model

The CSCI object model is shown in a number of views; Figures 4.3-1 to 4.3-8. Each view aims to capture a different aspect of the CSCI capabilities. Classes will appear in several views, only the operations and attributes pertinent to the capability being illustrated are included in the figures in order to reduce confusion. Complete class descriptions are provided in text in section 4.4.

4.3.1 PGE Profile View

The PGE profile view (Figure 4.3-1) describes the classes which are provided the Algorithm Integration and Test Tools to allow the entry of information that describes a PGE to the Production Planning CSCI.

The key classes introduced within this model are:

- PlPGEProfile: This class describes the collection of information that describes a PGE to the Planning subsystem.
- PlPGE: This is the base class within a generalization hierarchy that describes PGEs. The class defines abstract operations required for the planning subsystem to determine when a PGE needs to be scheduled, as well as containing the key attributes defining the PGE.
- PlDataScheduled: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined directly from the period of some primary input data type.
- PlTimeScheduled: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc.
- PlDataType: This class describes the data types used or output from PGEs. The PlDataType class can be thought of as a proxy to the Earth Science Data Type of the Data Server. The class contains the information required by the Planning subsystem to describe the inputs and outputs of a PGE.
- PlDataTypeReq: This describes the correlation between a PGE and the input data type. This class contains the data that allows a PGE to select the required granules of a given input type to perform a production task.

- PlOutputYield: Specifies the recipe to describe the output data granules for a PGE.
- PlUserParameters: Describes any user defined parameters that have to be associated to a PGE.
- PlResourceRequirements: This class contains a description of the resource requirements of a PGE, which may be matched against the resource configuration known to the Planning subsystem.
- PlPerformance: This class describes the performance statistics of a PGE. These performance statistics are established at AI&T. The class also contains attributes to describe the statistics updated from the Data Processing subsystem.

The interactions of these classes are described within the following scenarios:

- 4.5.1 PGE Profile

4.3.2 Production Request View

This view (Figure 4.3-2) describes the classes used during the entry of a production request and the subsequent process by which the Production Request is translated into Data Processing Requests.

The following key classes are introduced:

- PlProductionRequest: A production request describes an order for data sets. A production request typically specifies a request for a Data Set to be produced for an extended period of time (e.g a month's worth of some product).
- PlDPR: The DPR describes a single run of a PGE and has associations to the data items required for that Data Processing Request.
- PlDataGranule: This class describes a data item to the planning subsystem. This object is in effect a proxy to the real data items within the ECS, the class describes the information that is needed in the Planning subsystem to allow selection of that granule by a PGE for a Data Processing Request. The PlDataGranule describes either data that is actually in the system or a prediction of data that will come into the system.
- PlDataSource: This class is the base class that provides the methods for predicting when external data will arrive within the ECS, for example, from SDPF or National Oceanic and Atmospheric Administration (NOAA). The Data Sources are specialized to describe the different ways that predictions are obtained.
- PlRoutineArrival: This class is a specialization of the PlDataSource class and describes the most frequent method for predicting data arrivals within the ECS (at least for the TRMM data sets). This class contains the attributes and operations required to describe routine ingest of external data.
- DsClQuery: This class is used to construct a query to the Data Server for information on data that already exists in the system and is required for reprocessing.
- DsClESDTReferenceCollector: When applied to the data server a query returns a list of ESDTs that fulfill the criteria of the query.
- DsClESDTReference: This class is used to extract the metadata for a particular granule returned from the Query.

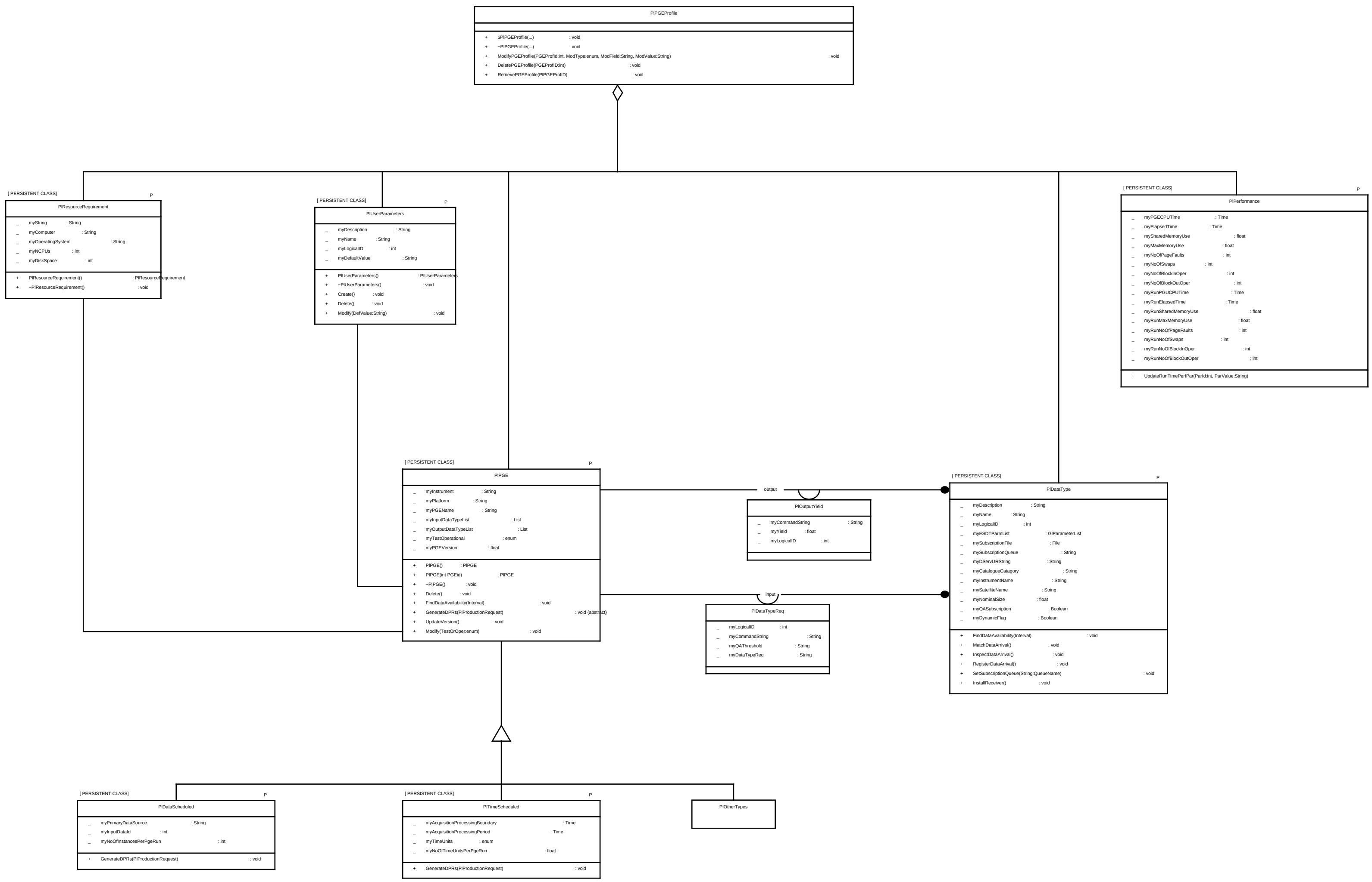


Figure 4.3-1. PGE Profile Object Model

The interactions of these classes are described within the following scenarios:

- 4.5.2 Production Request
- 4.5.3 Data Availability

4.3.3 Subscription Submission View

This view (Figure 4.3-3) describes the classes used during the generation, submission and withdrawal of Subscriptions for the notification of data arrival. The diagram essentially describes the key components of the subscription submission application.

The classes that are introduced in this model are:

- PlSubscriptionSubmitIF: This class is an abstraction for the user interface to the subscription submission application. The interface will be developed with a suitable GUI builder tool. See section 4.6.3 for a fuller description of this user interface.
- IoAdAdvertisingSrv_C: The advertising service provides access to the advertisement collections.
- IoAdServiceCollection_C: The advertising collection provides access to the ECS advertisements, which describe the services available for the ECS data. The advertisements of interest to the Planning subsystem are those that permit subscribing to insert events within the Science Data Server.
- IoAdServiceAdvertisement_C: The advertisement is used in order to build the Data Server Subscription
- DsClSubscription: This class is used to register and withdraw subscription in the Data Server. The constructor for this class is overloaded to permit the specification of the callback queue on which the generator of the subscription is notified of events which fulfill the subscription.
- MsUsProfile: The user profile describes the originator of the subscription to the Data Server.
- PlDataType: The status of the subscription is recorded within the planning Data Type abstraction through modification of the subscription queue attribute.

The interactions of these classes are described within the following scenarios:

- 4.5.4 Submitting a subscription
- 4.5.5 Withdrawing a subscription

4.3.4 Production Planning View

This view (Figure 4.3-4) describes the classes used in the creation and management of plans. The design presented here for a number of key classes is at a high level of abstraction. This level of abstraction is offered to explain the design without going into detail about the planning framework (Delphi) which supports the plan generation. The planning framework within the Production Planning CSCI is documented in detail within the Planning Object Library CSC, section 4.6.5. That section contains object models, class descriptions and scenarios for the framework and maps the objects used in framework to the abstractions presented here.

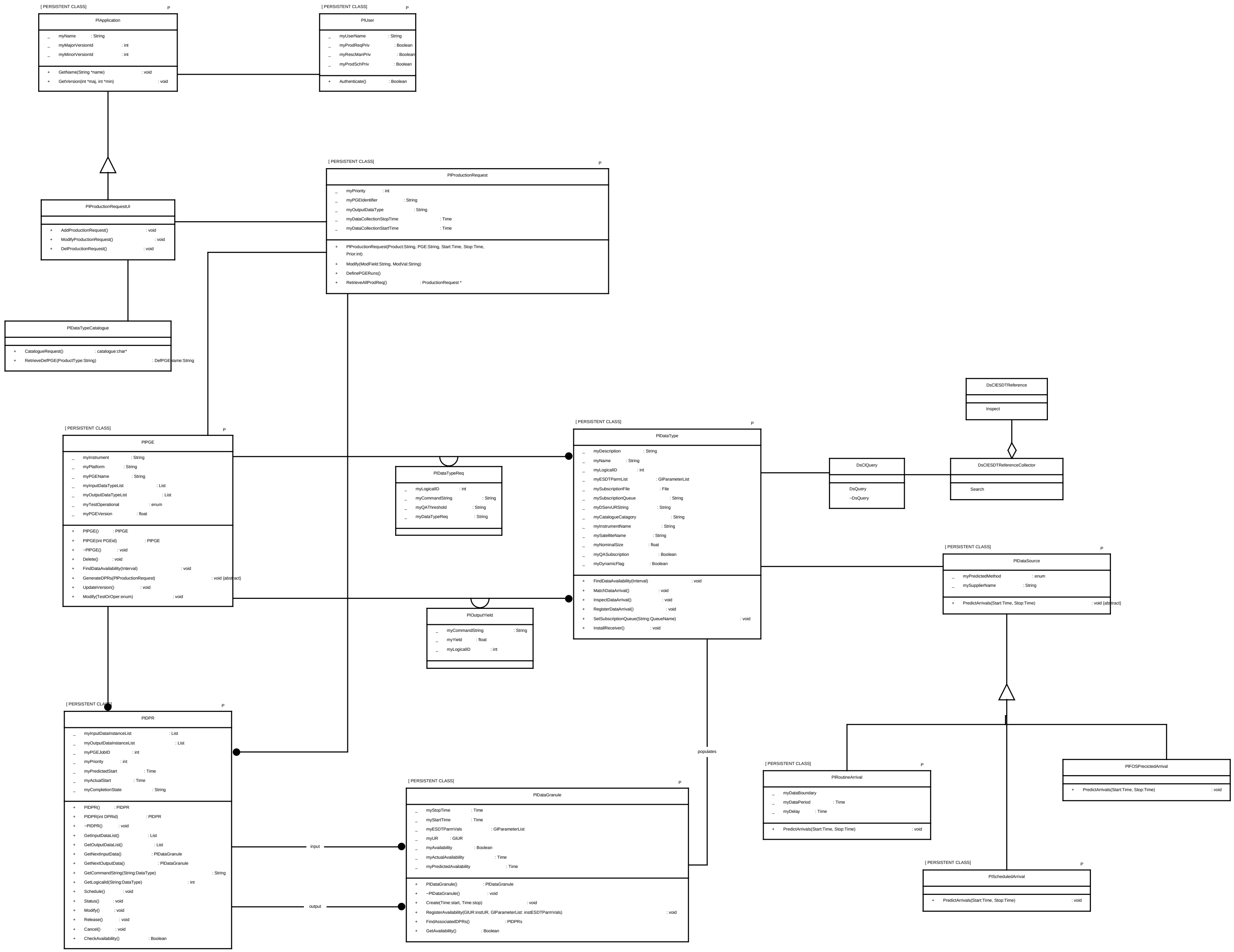


Figure 4.3-2. Production Request Object Model

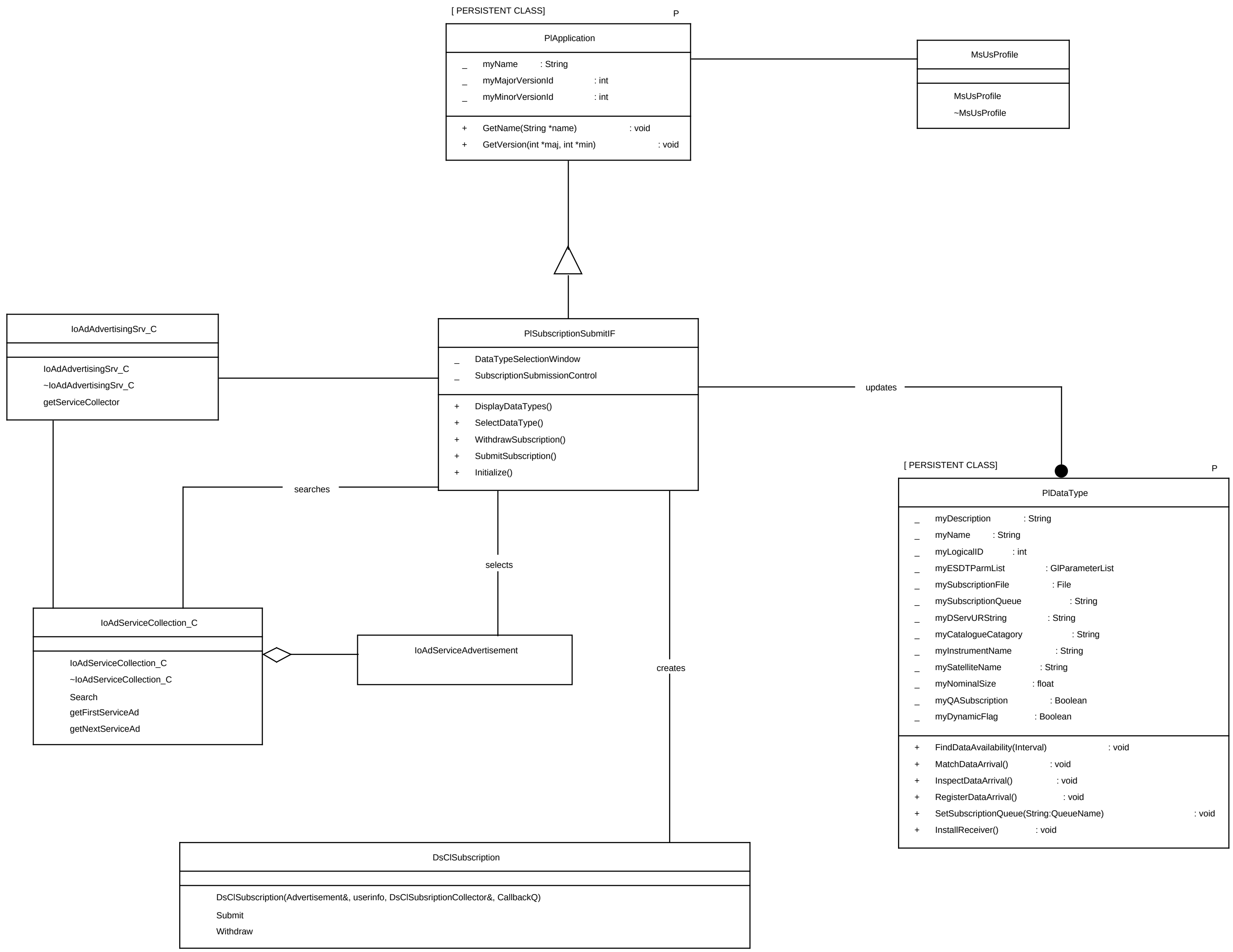


Figure 4.3-3. Subscription Submission Object Model

The diagram introduces the following key classes:

- PlPlanningWorkbenchUI: This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool.
- PlPlan: This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.
- PlResourceManager: This class represents an abstraction for the resource management capabilities used when generating a plan, describing the operations required to match resource requirements of an activity to the available resources, and to allocate the resource for the activity.
- PlResourceRequiremet: This class describes the resource requirements for a task. These requirements may then be matched against the actual resources available.
- PlActivity: This class describes an item within a plan. The activity class is a base class within a specialization hierarchy describing the different activities which occur in the production plan.
- PlPGEActivity: This class is a specialization of the PlActivity class. The class describes a Data Processing Request - a run of a PGE - within the plan.
- PlGroundActivity: This class is a generalization of the PlActivity class. The class describes a Ground Event within the plan.
- PlGroundEvent: This class describes a Ground Event which is recorded in the PDPS database. A Ground Event marks the allocation of resources to some non-production task such as maintenance.
- PlPublishedPlan: This object encapsulates the methods required to insert externalized formats of the plan into the document data server.

The interactions of these classes are described within the following scenarios:

- 4.5.6 Adding/Modify/Deleting a Ground Event
- 4.5.7 Creating a Plan
- 4.5.8 Deleting a Plan
- 4.5.9 Publishing a Plan

4.3.5 Resource Management View

This view (Figure 4.3-5) describes the classes used in the description of the production resource configuration. The MSS provides the planning subsystem (and all subsystems) with resource configuration information. The view is a continuation from the Production Planning Workbench view, presented separately to reduce the complexity of the Workbench diagram. The details for how the resource configuration is integrated into the framework for the production planning workbench are presented in section 4.6.5.

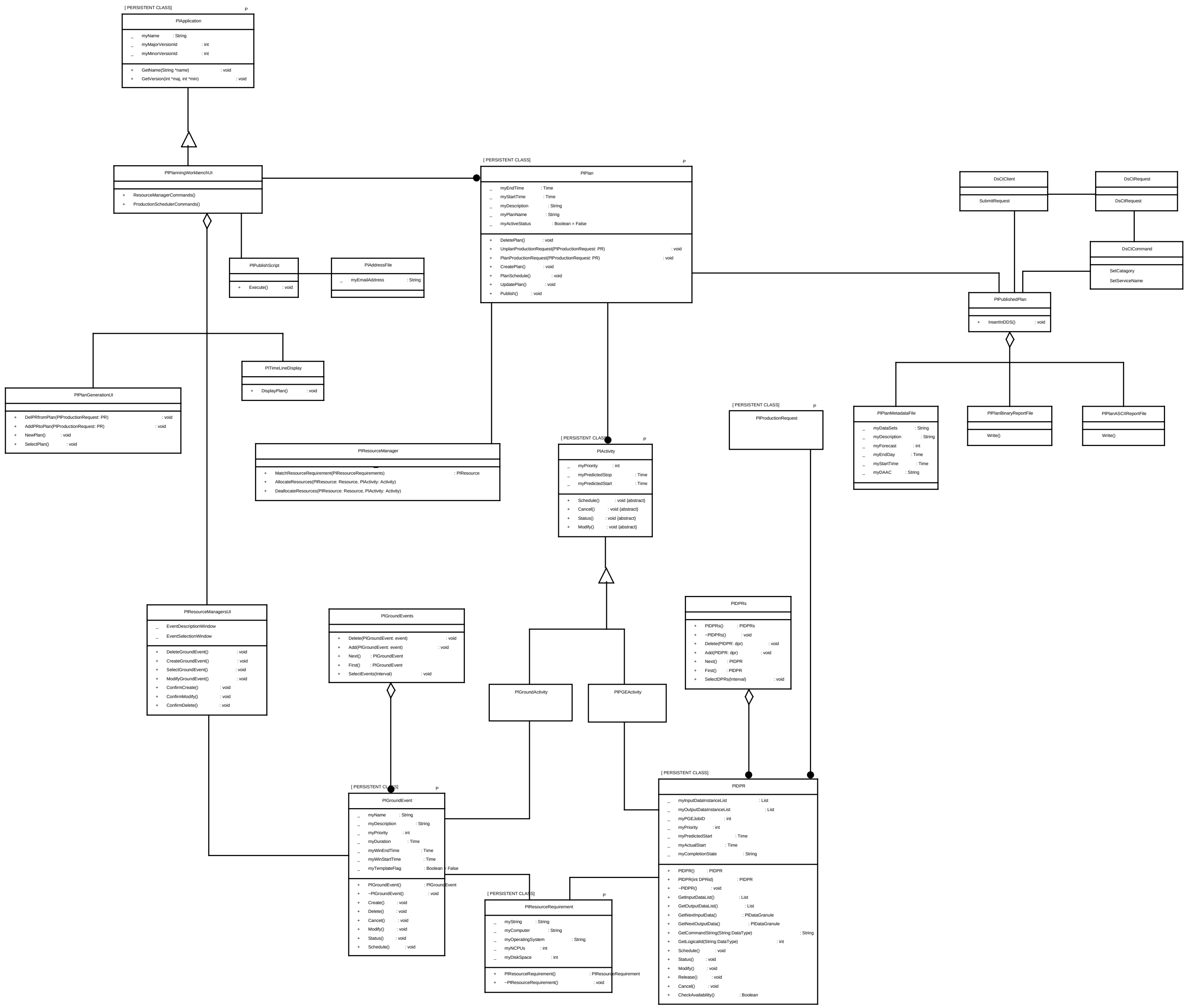


Figure 4.3-4. Production Planning Object Model

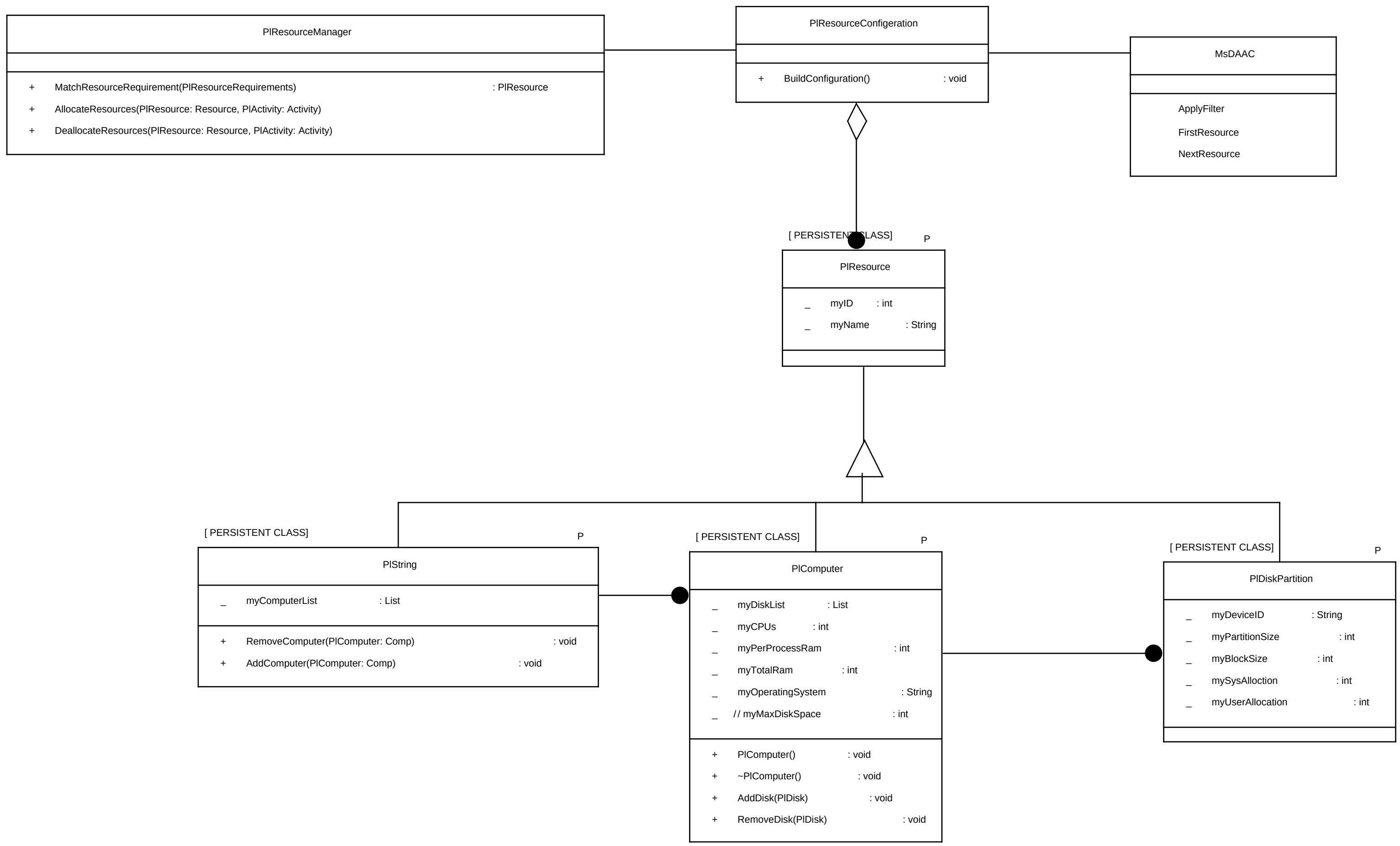


Figure 4.3-5. Resource Management Object Model

The diagram introduces the following key classes:

- PlResource: This class is the base class in a generalization heirachy describing the production resources.
- PlString: A string describes the logical collection of a number of resources allocated for an instruments processing needs.
- PlComputer: This class describes the production computers
- PlDiskPartition: This class describes the disk resources for data production.
- MsDAAC: This class provides MSS configuration information to the planning subsystem.

The interactions of these classes are described in the following scenario

- 4.5.10 Building the Resource Configuration

4.3.6 Plan Activation View

This view (4.3-6) again shows more detail for the Planning Workbench. It explicitly shows the Data Processing subsystem interface.

The diagram introduces the following key classes:

- PlProductionPlannersUI: This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool. The class does describe the basic operations that are provided from the interface.
- PlActivePlan: This class is the specialization of the PlPlan class and contains the methods to manage the activation, cancelling, and statesman of a plan.
- DpPrScheduler: This class is the interface to the Data Processing subsystem for entering jobs into the job scheduler.

The interactions of these classes are described in the following scenario:

- 4.5.11 Activating a Plan,
- 4.5.12 Cancelling a Plan, and
- 4.5.13 Statusing/Updating a Plan.

4.3.7 Subscription Manager View

This view (4.3-7) describes the classes used in the management of notification that a subscription has been fulfilled.

This view introduces the following key classes:

- PlSubscriptionManager: This class contains the main application methods associated with the subscription manager application
- PlSubscriptionManagerCallbacks: Specialization class to provide the subscription manager callbacks to the MsManager in order to be activated for the appropriate lifecycle services (shutdown, resume, suspend notify).
- MsManager: This class contains the external interfaces required by the MSS agents in order to activate the appropriate lifecycle services.

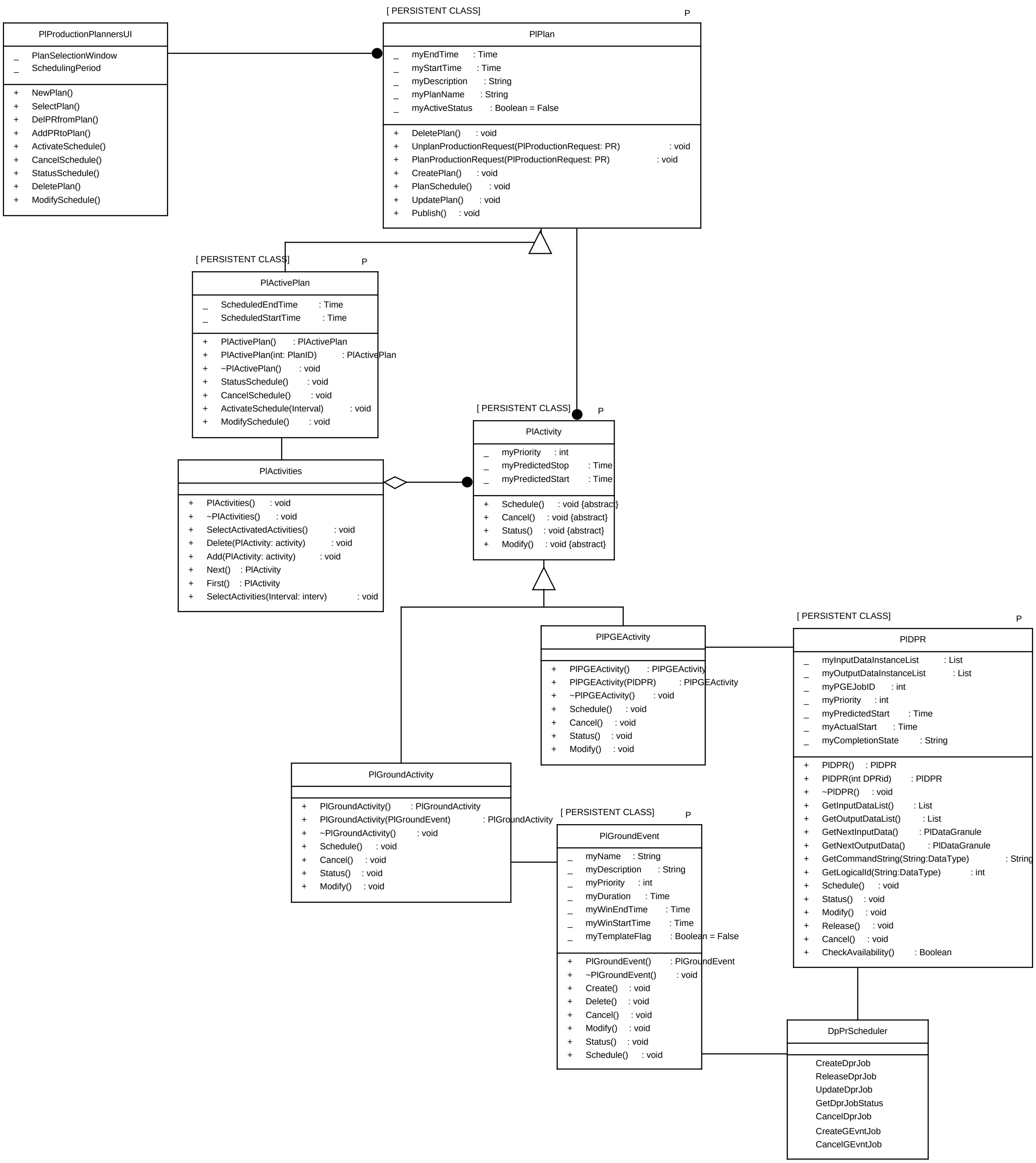


Figure 4.3-6. Plan Activation Object Model

- EcMpMsgPsngCtrl: This class contains the communications services for guaranteed asynchronous notification used for subscription notification
- GlUR: This class is used to create a Universal Reference (UR) from the Data Server subscription notification
- DsClESDTReference: This class provides the subscription manager with services by which to access the metadata associated to a Universal Reference.

The interactions of these classes are described within the following scenarios:

- 4.5.14 Subscription Manager Startup
- 4.5.15 Subscription Notification
- 4.5.16 Subscription Manager Shutdown

4.3.8 DBMS Proxy Agent View

This view (Figure 4.3-8) describes the classes used in the management of the lifecycle services for the PDPS DBMS. This application is almost a template reuse of the MSS services for Proxy Agents. For further details of the MSS agents and proxy agents refer to Release A CSMS Management Subsystem Design Specification for the ECS Project.

4.4 Class Descriptions

4.4.1 DpPrScheduler Class

DpPrScheduler: This class is the interface to the Data Processing subsystem for entering jobs into the job scheduler.

The full description for this class is provided in the Data Processing Design Specification.

4.4.2 DsClESDTReference Class

DsClESDTReference: This class provides the subscription manager with services by which to access the metadata associated to a Universal Reference.

The full description for this class is provided in the Data Server Design Specification.

4.4.3 DsClESDTReferenceCollector Class

DsClESDTReferenceCollector: When applied to the data server a query returns a list of ESDTs that fulfill the criteria of the query.

The full description for this class is provided in the Data Server Design Specification.

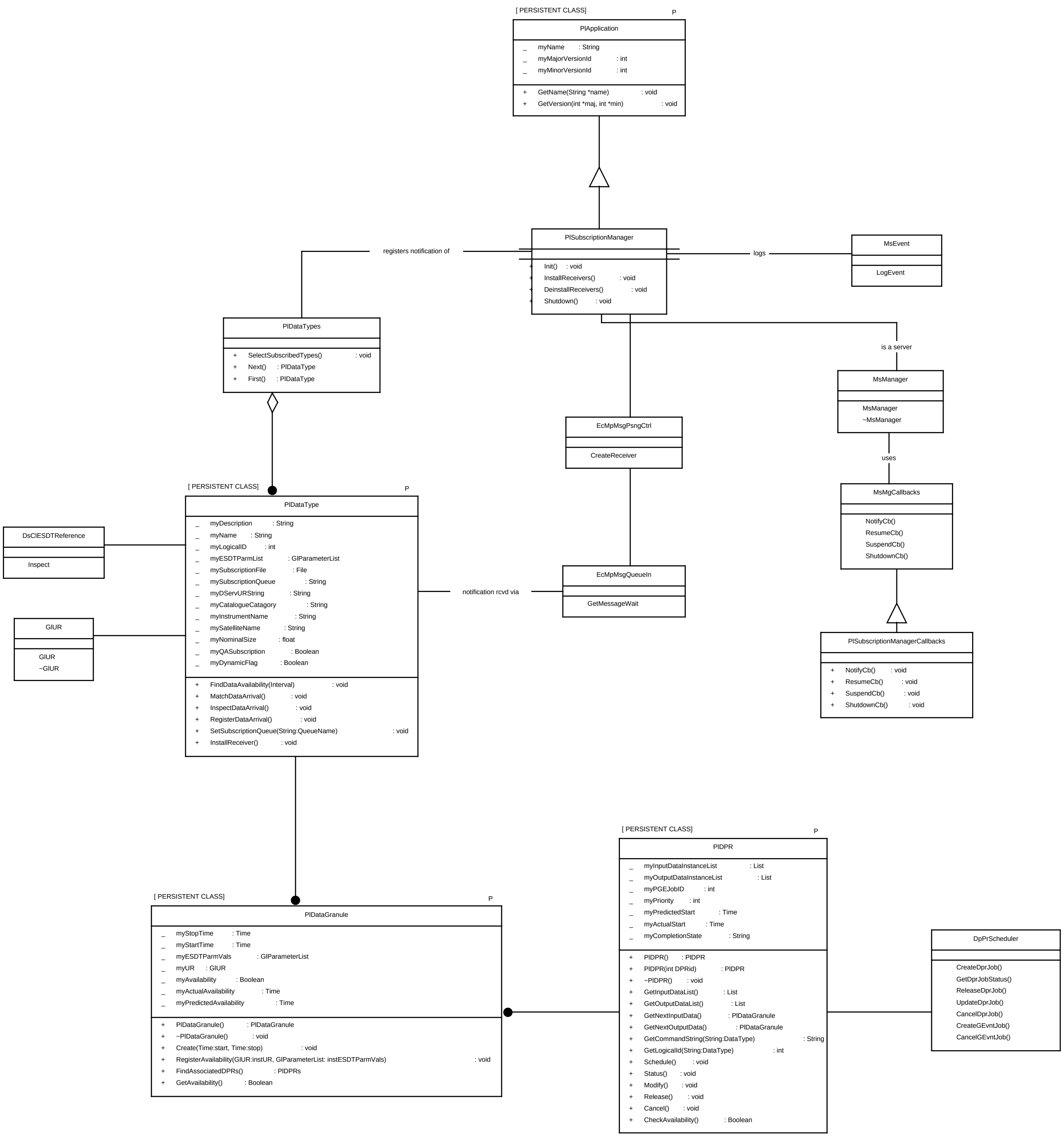


Figure 4.3-7. Subscription Manager Object Model

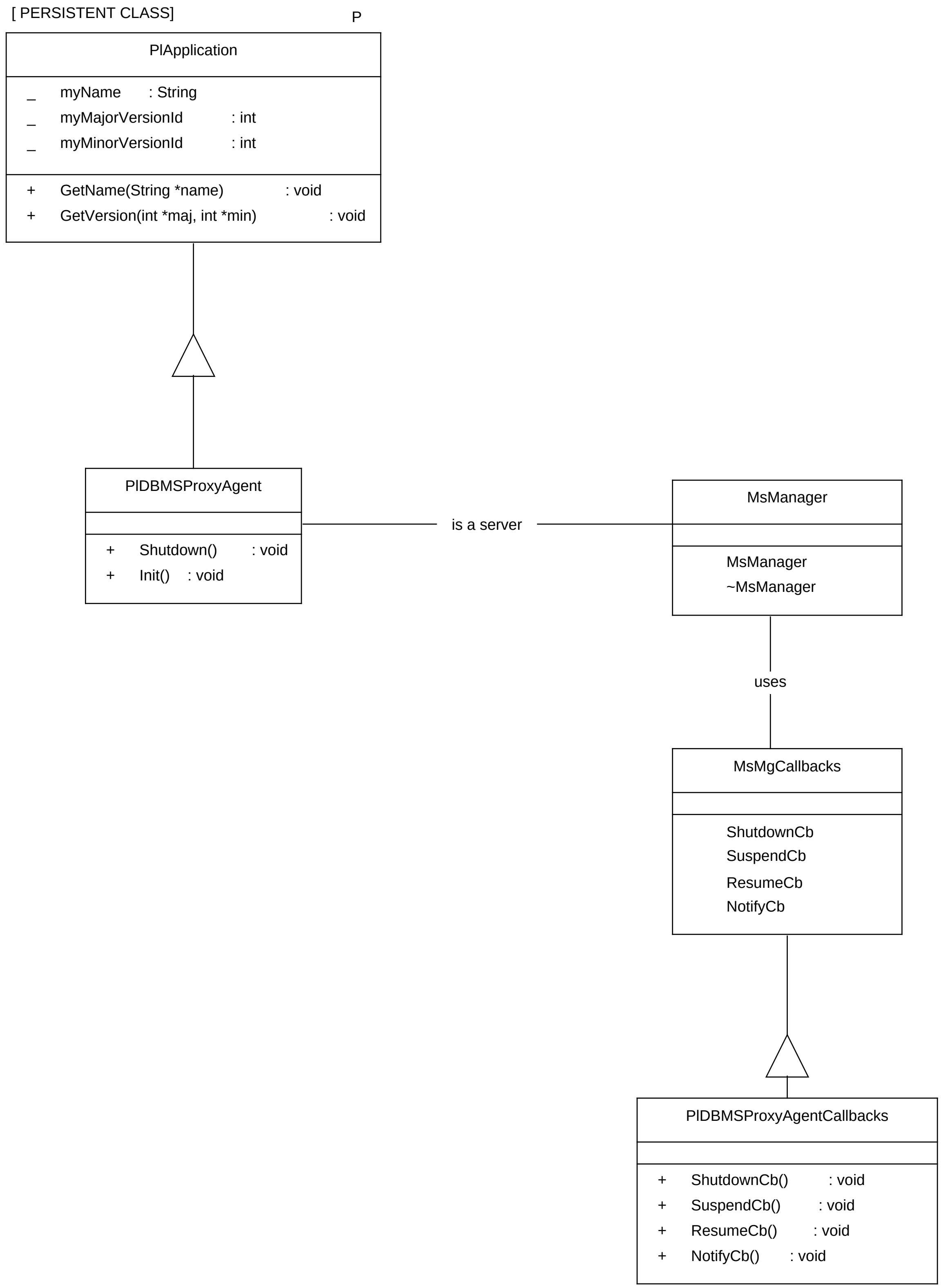


Figure 4.3-8. DBMS Proxy Agent Object Model

4.4.4 DsClQuery Class

DsClQuery: This class is used to construct a query to the Data Server for information on data that already exists in the system and is required for reprocessing.

The full description for this class is provided in the Data Server Design Specification.

4.4.5 DsClSubscription Class

DsClSubscription: This class is used to register and withdraw subscription in the Data Server. The constructor for this class is overloaded to permit the specification of the callback queue on which the generator of the subscription is notified of events which fulfill the subscription.

The full description for this class is provided in the Data Server Design Specification.

4.4.6 DsCtClient Class

DsCtClient: This class is provided by the Document Data Server, and allows the Planning Subsystem to submit a request for the insert of a Plan.

The full description for this class is provided in the Data Server Design Specification.

4.4.7 DsCtCommand Class

DsCtCommand: A command is used within a request to the Document Data Server, this command is built with a parameter list syntax to specify the command.

The full description for this class is provided in the Data Server Design Specification.

4.4.8 DsCtRequest Class

DsCtRequest: A DsCtCommand command is built into a request to the Document Data Server.

The full description for this class is provided in the Data Server Design Specification.

4.4.9 EcMpMsgPsngCtrl Class

EcMpMsgPsngCtrl: This class contains the communications services for guaranteed asynchronous notification used for subscription notification.

The full description for this class is provided in the Communications Subsystem Design Specification.

4.4.10 EcMpMsgQueueIn Class

EcMpMsgQueueIn: This class contains the communications services for guaranteed asynchronous notification used for subscription notification

The full description for this class is provided in the Communications Subsystem Design Specification.

4.4.11 GlUR Class

GlUR: This class is used to create a Universal Reference (UR) from the Data Server subscription notification

The full description for this class is provided in the internal interfaces document.

4.4.12 IoAdAdvertisingSrv_C Class

IoAdAdvertisingSrv_C: This class is used to obtain the Advertising Collection from the Advertising Server.

The full description for this class is provided in the Interoperability Subsystem Design Specification.

4.4.13 IoAdServiceAdvertisement Class

IoAdServiceAdvertisement_C: The advertisement is used in order to build the Data Server Subscription.

The full description for this class is provided in the Interoperability Subsystem Design Specification.

4.4.14 IoAdServiceCollection_C Class

IoAdServiceCollection_C: The advertising collection provides access to the ECS advertisements, which describe the services available for the ECS data. The advertisements of interest to the Planning subsystem are those that permit subscribing to insert events within the Science Data Server.

The full description for this class is provided in the Interoperability Subsystem Design Specification.

4.4.15 MsDAAC Class

MsDAAC: This class provides MSS configuration information to the planning subsystem.

The full description for this class is provided in the Management Subsystem Design Specification.

4.4.16 MsEvent Class

MsEvent: This class is used to log management information to the MSS.

The full description for this class is provided in the Management Subsystem Design Specification.

4.4.17 MsManager Class

MsManager: This class contains the external interfaces required by the MSS agents in order to activate the appropriate lifecycle services.

The full description for this class is provided in the Management Subsystem Design Specification.

4.4.18 MsMgCallbacks Class

MsMgCallbacks: This is a base class for specialization of callbacks required to interface to the MSS lifecycle services.

The full description for this class is provided in the Management Subsystem Design Specification.

4.4.19 MsUsProfile Class

MsUsProfile: The user profile describes the originator of the subscription to the Data Server.

The full description for this class is provided in the Management Subsystem Design Specification.

4.4.20 PlActivePlan Class

Parent Class: PlPlan
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class is the specialization of the PlPlan class and contains the methods to manage the activation, cancelling, and statusing of a plan

Attributes:

ScheduledEndTime - The end date and time for the portion of the plan entered into the data processing subsystem scheduler.
Data Type: Time
Privilege: Private
Default Value:

ScheduledStartTime - The start date and time for the portion of the plan entered into the data processing subsystem scheduler.
Data Type: Time
Privilege: Private
Default Value:

Operations:

ActivateSchedule - Schedule the activities within the specified interval in the plan with the data processing scheduler.
Arguments: Interval
Return Type: void
Privilege: Public
PDL:{
// Create an ordered list of activities which are within the plan
// and lie within the interval specified

// Iterate through the list, and invoke the schedule method for
// each activity
}

CancelSchedule - Cancel the activities that have been scheduled from the plan within the data processing scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler

// Iterate through the list, and invoke the cancel method for
// each activity
}

ModifySchedule - Modify the activities that have been scheduled from the plan within the data processing scheduler. The attributes of the activity that may be modified are priority and the anticipated start / stop times of the activity used to set the alarms in the data processing scheduler. Any greater modification should be managed by canceling the activity and scheduling a new activity.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler

// Iterate through the list, and invoke the modify method for
// each activity
}

PlActivePlan - Creator method for the active plan
Arguments:
Return Type: PlActivePlan
Privilege: Public

PlActivePlan - Recreates the active plan from the DBMS given an identifier for that plan.
Arguments: int: PlanID
Return Type: PlActivePlan
Privilege: Public

StatusSchedule - Status the plan against the schedule active in the data processing scheduler.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler

// Iterate through the list, and invoke the status method for
// each activity -- this is reflected within the activity object
// and DBMS

// Update the plan accordingly
}

~PlActivePlan - Destructor method for the active plan.
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlActivePlan class has associations with the following classes:
Class: PlActivities

4.4.21 PlActivities Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class is a container class for activities in the PDPS database. This class may be implemented by a suitable Rogue Wave template class.

Attributes:

None

Operations:

Add - Add an activity to the collection
Arguments: PlActivity: activity
Return Type: void
Privilege: Public

Delete - Delete an activity from the collection.
Arguments: PlActivity: activity
Return Type: void
Privilege: Public

First - Returns the first activity within the collection.
Arguments:
Return Type: PlActivity
Privilege: Public

Next - Returns the next activity within the collection (or NULL if no more activities).
Arguments:
Return Type: PlActivity
Privilege: Public

PlActivities - Constructor for the container class.
Arguments:
Return Type: void
Privilege: Public

SelectActivatedActivities - Select those activities within the plan that have been activated, i.e. those scheduled within the data processing scheduler.
Arguments:
Return Type: void
Privilege: Public

SelectActivities - Builds the collection for all activities within the plan that lies within the time interval specified.
Arguments: Interval: interv
Return Type: void
Privilege: Public

~PlActivities - Destructor method.
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlActivities class has associations with the following classes:
Class: PlActivePlan

4.4.22 PlActivity Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes an item within a plan. The activity class is a base class within a specialization heirachy descrining the different activities which occur in the production plan.

Attributes:

myPredictedStart - The predicted start time of an activity
Data Type: Time
Privilege: Private
Default Value:

myPredictedStop - The predicted end time of an activity.
Data Type: Time
Privilege: Private
Default Value:

myPriority - Priority of the activity within the plan.
Data Type: int
Privilege: Private
Default Value:

Operations:

Cancel - Abstract operation for the cancelling of an activity within the Data Processing subsystem.
Arguments:
Return Type: void
Privilege: Public
This is an abstract operation

Modify - Abstract operation for the modification of an activity within the Data Processing subsystem.
Arguments:
Return Type: void
Privilege: Public
This is an abstract operation

Schedule - Abstract operation for the scheduling of an activity within the Data Processing subsystem.
Arguments:
Return Type: void
Privilege: Public
This is an abstract operation

Status - Abstract operation for the statusing of an activity from the Data Processing subsystem.
Arguments:
Return Type: void
Privilege: Public
This is an abstract operation

Associations:

The PlActivity class has associations with the following classes:
Class: PlPlan

4.4.23 PlAddressFile Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class represents a list of recipients to be informed when a new plan is published. The object oriented notation is used so that this element can be captured in the design, however this class will probably be implemented by a simple text file to be read by the publish script. These components will be replaced when the document data server supports subscriptions to published plans.

Attributes:

myEmailAddress - Email address for recipient to be informed of a new script
Data Type: String
Privilege: Private
Default Value:

Operations:

None

Associations:

The PlAddressFile class has associations with the following classes:
Class: PlPublishScript

4.4.24 PlApplication Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
A base class for application programs describing name, version etc.

Attributes:

myMajorVersionId - Application major version number
Data Type: int
Privilege: Private
Default Value:
This is a Constant.

myMinorVersionId - Software minor version id
Data Type: int
Privilege: Private
Default Value:
This is a Constant.

myName - Application name
Data Type: String
Privilege: Private
Default Value:
This is a Constant.

Operations:

GetName - Operation to retreive the application name
Arguments: String *name
Return Type: void
Privilege: Public

GetVersion - Operation to retrieve the application version number
Arguments: int *maj, int *min
Return Type: void
Privilege: Public

Associations:

The PlApplication class has associations with the following classes:
Class: PlUser

4.4.25 PlComputer Class

Parent Class: PlResource

Public: NoDistributed Object: No

Persistent Class: True

Purpose and Description:

This class describes the computers which are part of the resource configuration for the production system.

Attributes:

/ myMaxDiskSpace - Derived max disk space

Data Type: int

Privilege: Private

Default Value:

This is a Derived Attribute.

myCPUs - The number of CPUs within the computer

Data Type: int

Privilege: Private

Default Value:

myDiskList - Describes the disks associated with the computer.

Data Type: List

Privilege: Private

Default Value:

myOperatingSystem - The operating system name and version

Data Type: String

Privilege: Private

Default Value:

myPerProcessRam - The operating system's allocation of ram per process

Data Type: int

Privilege: Private

Default Value:

myTotalRam - The total Ram for the computer

Data Type: int

Privilege: Private

Default Value:

Operations:

AddDisk - Add a disk to the list
Arguments: PlDisk
Return Type: void
Privilege: Public

PlComputer - Class constructor
Arguments:
Return Type: void
Privilege: Public

RemoveDisk - remove a disk from the list
Arguments: PlDisk
Return Type: void
Privilege: Public

~PlComputer - Class destructor
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlComputer class has associations with the following classes:
Class: PlDiskPartition
Class: PlString

4.4.26 PlDBMSProxyAgent Class

Parent Class: PlApplication
Public: NoDistributed Object: No
Purpose and Description:
This class describes the main body for the proxy agent required to manage the lifecycle services of the PDPS DBMS. The proxy agent will be an almost template re-use of the MSS capabilities.

Attributes:

All Attributes inherited from parent class

Operations:

Init - Initializes the proxy agent.
Arguments:
Return Type: void
Privilege: Public

Shutdown - Shuts down the PDPS database server.
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDBMSProxyAgent class has associations with the following classes:
Class: MsManager isaserver

4.4.27 PlDBMSProxyAgentCallbacks Class

Parent Class: MsMgCallbacks
Public: NoDistributed Object: No
Purpose and Description:
Specialization class to provide the DBMS proxy agent callbacks to the MsManager in order to be activated for the appropriate lifecycle services (shutdown, resume, suspend notify).

Attributes:

All Attributes inherited from parent class

Operations:

NotifyCb - Notify callback not used at Release A. A null method will be provided.
Arguments:
Return Type: void
Privilege: Public

ResumeCb - Resume not supported at Release A.
Arguments:
Return Type: void
Privilege: Public

ShutdownCb - Shutdown invokes the DBMS proxy agent shutdown method, which invokes the appropriate sybase shutdown procedure.
Arguments:
Return Type: void
Privilege: Public

SuspendCb - Suspend not supported at Release A
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDBMSProxyAgentCallbacks class has associations with the following classes:
None

4.4.28 PlDPR Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes an individual run of a PGE.

Attributes:

myActualStart - Actual start time from data processing
Data Type: Time
Privilege: Private
Default Value:

myCompletionState - Status indicator describing active status of data processing request
Data Type: String
Privilege: Private
Default Value:

myInputDataInstanceList - List describing all the input files required within for the PGE
Data Type: List
Privilege: Private
Default Value:

myOutputDataInstanceList - List describing all the output files to be produced by the PGE
Data Type: List
Privilege: Private
Default Value:

myPGEJobID - Identifier returned from the data processing subsystem to define the job
Data Type: int
Privilege: Private
Default Value:

myPredictedStart - Predicted start time from plan activity
Data Type: Time
Privilege: Private
Default Value:

myPriority - Priority for the data processing request is inherited from the production reqeust, but may be modified individually
Data Type: int
Privilege: Private
Default Value:

Operations:

Cancel - Cancel the DPR within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the cancel DPR job method of the DpPrJobScheduler
}

CheckAvailability - Checks to see if all the data dependies for a DPR have been fulfilled. Returns true or false.
Arguments:
Return Type: Boolean
Privilege: Public

PDL:{
// set availability TRUE

// iterate through the input data list

// check the availability of the granule

// if data not avaialble set availability FALSE

// return availability
}

GetCommandString - returns the command string for the given data type, the command string is used to build the data server insert / aquire request
Arguments: String:DataType
Return Type: String
Privilege: Public

GetInputDataList - Returns a pointer to the list of input data granules
Arguments:
Return Type: List
Privilege: Public

GetLogicalId - returns the logical identifier for the given data type. the logical identifier is used to identify the data type within the science software and is requried for the construction of the process control info file
Arguments: String:DataType
Return Type: int
Privilege: Public

GetNextInputData - Returns the next input data granule in the list
Arguments:
Return Type: PlDataGranule
Privilege: Public

GetNextOutputData - returns the next output data granule in the list
Arguments:
Return Type: PlDataGranule
Privilege: Public

GetOutputDataList - Returns a pointer to the list of output data granules
Arguments:
Return Type: List
Privilege: Public

Modify - Modify the DPR within the data processing job scheduler. The attributes of the DPR that may have been modified are priority and the anticipated start / stop times of the DPR used to set the alarms in the data processing scheduler.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the modify DPR job method of the DpPrJobScheduler
}

PlDPR - Constructor method
Arguments:
Return Type: PlDPR
Privilege: Public

PlDPR - Overloaded constructor to recreate DPR from the PDPS DBMS.
Arguments: int DPRid
Return Type: PlDPR
Privilege: Public

Release - Releases the DPR within the data processing job scheduler. The DPR is released when the data dependencies of that DPR have been fulfilled.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the release DPR job method of the DpPrJobScheduler
}

Schedule - schedule the DPR within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the create DPR job method of the DpPrJobScheduler
}

Status - status the DPR within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{

// calls the status DPR job method of the DpPrJobScheduler
}

~PlDPR - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDPR class has associations with the following classes:
Class: PlPGEActivity
Class: PlProductionRequest
Class: PlResourceRequirement
PlDPRs (Aggregation)

4.4.29 PlDPRs Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Purpose and Description:
This is a collector class for the PlDPR class, and contains methods to select Data Processing Requests from the PDPS database and to iterate through them. This class may be implemented by a suitable Rogue Wave template class.

Attributes:

None

Operations:

Add - Add a DPR to the collection
Arguments: PlDPR: dpr
Return Type: void
Privilege: Public

Delete - Delete an DPR from the collection.
Arguments: PlDPR: dpr
Return Type: void

Privilege: Public

First - Returns the first DPR within the collection
Arguments:
Return Type: PlDPR
Privilege: Public

Next - Returns the next DPR within the collection (or NULL if no more activities).
Arguments:
Return Type: PlDPR
Privilege: Public

PlDPRs - Constructor method
Arguments:
Return Type: PlDPRs
Privilege: Public

SelectDPRs - Builds the collection for all DPRs within the plan that lies within the time interval specified
Arguments: Interval
Return Type: void
Privilege: Public

~PlDPRs - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDPRs class has associations with the following classes:
None

4.4.30 PlDataGranule Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes individual instances or granules of data types.

Attributes:

myActualAvailability - Data and time that data was made available to subscription manager.
Data Type: Time
Privilege: Private
Default Value:

myAvailability - Flag to indicate availability of data.
Data Type: Boolean
Privilege: Private
Default Value:

myESDTParmVals - Selected metadata fields associated to the data type required to determine suitability in the production (such as quality info or geophysical attributes).
Data Type: GlParameterList
Privilege: Private
Default Value:

myPredictedAvailability - Predicted time at which the data will be available in ECS, used to determine PGE schedule estimates.
Data Type: Time
Privilege: Private
Default Value:

myStartTime - The start time and date of the data.
Data Type: Time
Privilege: Private
Default Value:

myStopTime - The stop time and date of the data.
Data Type: Time
Privilege: Private
Default Value:

myUR - Universal Reference by which to reference the granule within the data server.
Data Type: GlUR
Privilege: Private
Default Value:

Operations:

Create - Create a entry in the DBMS if this is a unique instance of the granule
Arguments: Time:start, Time:stop
Return Type: void
Privilege: Public

FindAssociatedDPRs - Method to determine the data processing requests associated to the data granule.
Arguments:
Return Type: PlDPRs
Privilege: Public

GetAvailability - Returns whether the data item is available within the ECS or not.
Arguments:
Return Type: Boolean
Privilege: Public

PlDataGranule - Constructor method
Arguments:
Return Type: PlDataGranule
Privilege: Public

RegisterAvailability - Method to register that a data instance that was predicted has arrived, and to record the UR and metadata associated to that data
Arguments: GlUR:instUR, GlParameterList: instESDTParmVals
Return Type: void
Privilege: Public

~PlDataGranule - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDataGranule class has associations with the following classes:
Class: PlDPR input
Class: PlDPR output
Class: PlDataType populates

4.4.31 PlDataGranule Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes individual instances or granules of data types.

Attributes:

myActualAvailability - Data and time that data was made available to subscription manager.
Data Type: Time
Privilege: Private
Default Value:

myAvailability - Flag to indicate availability of data.
Data Type: Boolean
Privilege: Private
Default Value:

myESDTParmVals - Selected metadata fields associated to the data type required to determine suitability in the production (such as quality info or geophysical attributes).
Data Type: GlParameterList
Privilege: Private
Default Value:

myPredictedAvailability - Predicted time at which the data will be available in ECS, used to determine PGE schedule estimates.
Data Type: Time
Privilege: Private
Default Value:

myStartTime - The start time and date of the data.
Data Type: Time
Privilege: Private
Default Value:

myStopTime - The stop time and date of the data.
Data Type: Time
Privilege: Private
Default Value:

myUR - Universal Reference by which to reference the granule within the data server.
Data Type: GlUR

Privilege: Private
Default Value:

Operations:

Create - Create a entry in the DBMS if this is a unique instance of the granule
Arguments: Time:start, Time:stop
Return Type: void
Privilege: Public

FindAssociatedDPRs - Method to determine the data processing requests associated to the data granule.
Arguments:
Return Type: PlDPRs
Privilege: Public

GetAvailability - Returns whether the data item is available within the ECS or not.
Arguments:
Return Type: Boolean
Privilege: Public

PlDataGranule - Constructor method
Arguments:
Return Type: PlDataGranule
Privilege: Public

RegisterAvailability - Method to register that a data instance that was predicted has arrived, and to record the UR and metadata associated to that data
Arguments: GlUR:instUR, GlParameterList: instESDTParmVals
Return Type: void
Privilege: Public

~PlDataGranule - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDataGranule class has associations with the following classes:
Class: PlDPR
Class: PlDataType

4.4.32 PlDataScheduled Class

Parent Class: PlPGE
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc.

Attributes:

myInputDataId - ID of the input data based on the arrival of which the PGE is scheduled to run
Data Type: int
Privilege: Private
Default Value:

myNoOfInstancesPerPgeRun - ID of the input data based on the arrival of which the PGE is scheduled to run
Data Type: int
Privilege: Private
Default Value:

myPrimaryDataSource - Identifies the source of the data
Data Type: String
Privilege: Private
Default Value:

Operations:

GenerateDPRs - Generate the data processing request that fulfill the production request
Arguments: PlProductionRequest
Return Type: void
Privilege: Public
PDL:{
// Select the primary input granules that are within the period of
// the production request

// Generate a DPR for each myNoOfInstancesPerPgeRun granules
// of input data

// Invoke the select data method for the DPR
}

Associations:

The PlDataScheduled class has associations with the following classes:
None

4.4.33 PlDataSource Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class is the base class that provides the methods for predicting when external data will arrive within the ECS from SDPF or NOAA for example. The Data Sources are specialized to describe the different ways that predictions are obtained.

Attributes:

myPredictedMethod - Describes the method by which the data availability prediction occurs - e.g., routine arrival, arrival at scheduled times, FOS based prediction.
Data Type: enum
Privilege: Private
Default Value:

mySupplierName - Identifies the supplier of the data.
Data Type: String
Privilege: Private
Default Value:

Operations:

PredictArrivals - Predicts data instance arrivals for a given period.
Arguments: Start:Time, Stop:Time
Return Type: void
Privilege: Public
This is an abstract operation

Associations:

The PlDataSource class has associations with the following classes:
Class: PlDataType

4.4.34 PlDataType Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes a data type known to the planning subsystem. This is a description of an input or output type, distinct to a granule or instance of the data type. The class is an abstraction or proxy that describes one of the Data Server ESDTs. The class captures data and operations that are required to subscribe and receive notification from the Data Server when a new instance of the Data Type arrives.

Attributes:

myCatalogueCatagory - Catalogue catagory indicates whether the Data Type is valid for a production request, which would indicate it is generated from a PGE at a site, otherwise the Data Type is some intermediate file, or input file received from another site.
Data Type: String
Privilege: Private
Default Value:

myDServURString - Universal Reference to identify Data Server providing services (retreive, insert, inspect) for the Data Type.
Data Type: String
Privilege: Private
Default Value:

myDescription - This attribute provides a text description for the Data Type
Data Type: String
Privilege: Private
Default Value:

myDynamicFlag - Indicates whether the Data Type is dynamic or static. Examples of dynamic are L0, L1 data sets etc. with a frequest update time. Examples of static are calibration files which only change with a new version of a PGE.
Data Type: Boolean
Privilege: Private
Default Value:

myESDTParmList - A parameter list used within the inspect to the Data Server, to retreive the metadata associated to a newly arrived instance of the Data Type.
Data Type: GlParameterList
Privilege: Private
Default Value:

myInstrumentName - Instrument name associated with PGE
Data Type: String
Privilege: Private
Default Value:

myLogicalID - Attribute describes the logical id by which this data type is indexed within a PGE
Data Type: int
Privilege: Private
Default Value:

myName - Attribute describes the ESDT Name for the data set within the Data Server.
Data Type: String
Privilege: Private
Default Value:

myNominalSize - Nominal size of the data type
Data Type: float
Privilege: Private
Default Value:

myQASubscription - Captures whether a subscription has been set up for the QA of this data type.
Data Type: Boolean
Privilege: Private
Default Value:

mySatelliteName - Satellite name associated to PGE
Data Type: String
Privilege: Private
Default Value:

mySubscriptionFile - File required by the receiver communications software to provide persistance store for the messages so that messages are captured in case of software failure.
Data Type: File
Privilege: Private
Default Value:

mySubscriptionQueue - A unique name associated to the communications queue used to receive subscription notification from the data server.
Data Type: String
Privilege: Private
Default Value:

Operations:

FindDataAvailability - Ensures that the Data Granules that are required to fulfill a production request are captured in the PDPS database
Arguments: Interval
Return Type: void
Privilege: Public
PDL:{
// For any period of the interval before the current time
// construct and submit query to the data server with the holdings
// for that data type

// Generate an instance of the PlDataGranule for each hit returned
// from the query, making sure that granule is not previously
// within the list

// For any period after the last recorded granule returned
// from the data server invoke the appropriate prediction
// method from the PlDataSource class
}

InspectDataArrival - Creates a UR from the notification message received and uses the UR to extract the requried metadata (ESDT parameter list) from the Data Server.
Arguments:
Return Type: void
Privilege: Public

InstallReceiver - Install a receiving queue in which to receive notification of subscription from the Data Server.
Arguments:
Return Type: void
Privilege: Public

MatchDataArrival - Match the data arrival with the predictions within Data Granule table.
Arguments:
Return Type: void
Privilege: Public

RegisterDataArrival - Procedure to be called on notification of data arrival, this a control procedure that manages the inspection of the data, comparison to the data instances, and then subsequent handling of associated DPRs to the arrived data.
Arguments:
Return Type: void
Privilege: Public

SetSubscriptionQueue
Arguments: String:QueueName
Return Type: void
Privilege: Public

Associations:

The PlDataType class has associations with the following classes:
Class: DsClQuery
Class: PlDataSource
Class: PlPGE
Class: PlDataGranule populates

4.4.35 PlDataTypeCatalogue Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class describes a collection that lists the data types that may be generated by the production system. The class is constructed from the PlDataTypes class using the catalogue catagory attribute from that class.

Attributes:

None

Operations:

CatalogueRequest - Request to retrieve all data types the ECS site is capable of generating.
Arguments:
Return Type: catalogue:char*
Privilege: Public

RetrieveDefPGE - Retrieves the default PGE and version corresponding to a product.
Arguments: ProductType:String
Return Type: DefPGEName:String
Privilege: Public

Associations:

The PlDataTypeCatalogue class has associations with the following classes:
Class: PlProductionRequestUI

4.4.36 PlDataTypeReq Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class contains the data that associates a PGE to an input data type.

Attributes:

myCommandString - This string is used to construct the acquire command for the data type, used to stage the data from the data server.
Data Type: String
Privilege: Private
Default Value:

myDataTypeReq - This attribute specifies the data type requirement of a PGE. The string specifies the select statement to be applied to the Data Granule table, to determine the input Data Granules for a given Data Processing Request.

Data Type: String
Privilege: Private
Default Value:

myLogicalID - The logical id relates to the PGE input identifier for a particular product type. This attribute is required within the SDP toolkit process control interface.
Data Type: int
Privilege: Private
Default Value:

myQAThreshold - This attribute describes the quality threshold to be applied to the data granules to "approve" their suitability for production before releasing a scheduled PGE. The threshold is specified in terms of the ESDT parameter list of the data type.
Data Type: String
Privilege: Private
Default Value:

Operations:

None

Associations:

The PlDataTypeReq class has associations with the following classes:
None

4.4.37 PlDataTypes Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
The class provides for a collection of Data Type objects selected from the PDPS database and suitable services to iterate over the collection. The class will be implemented using a suitable collection class from Rogue Wave.

Attributes:

None

Operations:

First - Returns the first Data Type from the collector
Arguments:
Return Type: PlDataType
Privilege: Public

Next - Returns the next Data Type from the collector
Arguments:
Return Type: PlDataType
Privilege: Public

SelectSubscribedTypes - Operation to initiate selection from PDPS database of DataTypes which have subscriptions to a Data Server.
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlDataTypes class has associations with the following classes:
Class: PlSubscriptionManager registersnotificationof

4.4.38 PlDiskPartition Class

Parent Class: PlResource
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes the disk resources for data production

Attributes:

myBlockSize - the block size for the partition
Data Type: int
Privilege: Private
Default Value:

myDeviceID - specifies the UNIX device id for the partition
Data Type: String

Privilege: Private
Default Value:

myPartitionSize - the total partition size
Data Type: int
Privilege: Private
Default Value:

mySysAlloction - the size of allocation of the partition for system usage
Data Type: int
Privilege: Private
Default Value:

myUserAllocation - the size of allocation of the partition for production usage
Data Type: int
Privilege: Private
Default Value:

Operations:

All Operations inherited from parent class

Associations:

The PlDiskPartition class has associations with the following classes:
Class: PlComputer

4.4.39 PlFOSPrecictedArrival Class

Parent Class: PlDataSource
Public: NoDistributed Object: No
Purpose and Description:
This object will not be developed for the TRMM release, it is shown as a placeholder for AM-1 when the data availability prediction may be built from the FOS APS schedules

Attributes:

All Attributes inherited from parent class

Operations:

PredictArrivals - Predicts data instance arrivals for a given period.
Arguments: Start:Time, Stop:Time
Return Type: void
Privilege: Public

Associations:

The PlFOSPrecictedArrival class has associations with the following classes:
None

4.4.40 PlGroundActivity Class

Parent Class: PlActivity
Public: NoDistributed Object: No
Purpose and Description:
This class is a generalization of the PlActivity class. The class describes a Ground Event within the plan.

Attributes:

All Attributes inherited from parent class

Operations:

Cancel - Cancel the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the cancel method of the Ground Event
}

Modify - Modify the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the modify method of the Ground Event

}

PlGroundActivity - Constructor method
Arguments:
Return Type: PlGroundActivity
Privilege: Public

PlGroundActivity - Create an activity from the Ground Event object
Arguments: PlGroundEvent
Return Type: PlGroundActivity
Privilege: Public

Schedule - Schedule the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the schedule method of the Ground Event
}

Status - Status the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the status method of the Ground Event
}

~PlGroundActivity - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlGroundActivity class has associations with the following classes:
Class: PlGroundEvent

4.4.41 PlGroundEvent Class

Parent Class: Not Applicable

Public: YesDistributed Object: No

Persistent Class: True

Purpose and Description:

This class describes a Ground Event which is recorded in the PDPS database. A Ground Event marks the allocation of resources to some none-production task such as maintainance.

Attributes:

myDescription - This attribute provides storage for a text description of a ground event

Data Type: String

Privilege: Private

Default Value:

This is a Constant.

myDuration - This attribute describes the duration of the Ground Event

Data Type: Time

Privilege: Private

Default Value:

This is a Constant.

myName

Data Type: String

Privilege: Private

Default Value:

myPriority - This attribute describes the priority of the Ground Event.

Data Type: int

Privilege: Private

Default Value:

This is a Constant.

myTemplateFlag - This attribute describes whether the ground event is to be saved as a template description of a ground events. Ground Events with this attribute True will be retained in the PDPS database. Those with this attribute as False will automatically be deleted a month after the completion date of the Ground Event.

Data Type: Boolean

Privilege: Private

Default Value: False

This is a Constant.

myWinEndTime - This attribute describes the end time of the window of opportunity for when the Ground Event may be planned
Data Type: Time
Privilege: Private
Default Value:
This is a Constant.

myWinStartTime - This attribute describes the start time of the window of opportunity for when the Ground Event may be planned
Data Type: Time
Privilege: Private
Default Value:
This is a Constant.

Operations:

Cancel - Cancel the Ground Event within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the cancel Ground Event method of the DpPrJobScheduler
}

Create - Create an ground event in the PDPS database
Arguments:
Return Type: void
Privilege: Public

Delete - Delete the ground event from the PDPS database
Arguments:
Return Type: void
Privilege: Public

Modify - Modify the Ground Event within the data processing job scheduler. The attributes of the Ground Event that may have been modified are priority and the anticipated start / stop times of the DPR used to set the alarms in the data processing scheduler.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the modify Ground Event job method of the DpPrJobScheduler
}

PlGroundEvent - Constructor method
Arguments:
Return Type: PlGroundEvent
Privilege: Public

Schedule - Schedule the Ground Event within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the create Ground Event job method of the DpPrJobScheduler
}

Status - Status the Ground Event within the data processing job scheduler
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the status Ground Event job method of the DpPrJobScheduler
}

~PlGroundEvent - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlGroundEvent class has associations with the following classes:
Class: DpPrScheduler
Class: PlGroundActivity

4.4.42 PlGroundEvents Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This is a collector class for the PlGroundEvent class, and contains methods to select Ground Events from the PDPS database and to iterate through them. This class may be

implemented by a suitable Rogue Wave template class

Attributes:

None

Operations:

Add - Add a Ground Event to the collection
Arguments: PlGroundEvent: event
Return Type: void
Privilege: Public

Delete - Delete a Ground Event from the collection
Arguments: PlGroundEvent: event
Return Type: void
Privilege: Public

First - Returns the first Ground Event within the collection.
Arguments:
Return Type: PlGroundEvent
Privilege: Public

Next - Returns the next Ground Event within the collection (or NULL if no more activities).
Arguments:
Return Type: PlGroundEvent
Privilege: Public

SelectEvents - Builds the collection for all Ground Events within the PDPS database that lie within the time interval specified.
Arguments: Interval
Return Type: void
Privilege: Public

Associations:

The PlGroundEvents class has associations with the following classes:
None

4.4.43 PlOutputYield Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
Specifies the recipe to describe the output data granules for a PGE.

Attributes:

myCommandString - This string is used to construct the insert command for the data type, used to destage the data to the data server.
Data Type: String
Privilege: Private
Default Value:

myLogicalID - The logical id relates to the PGE output identifier for a particular product type. This attribute is required within the SDP toolkit process control interface.
Data Type: int
Privilege: Private
Default Value:

myYield - Describes the number of data granules produced from the Data Processing Request. These are assumed to be evenly distributed in time accross the acquisition processing time of the DPR.
Data Type: float
Privilege: Private
Default Value:

Operations:

None

Associations:

The PlOutputYield class has associations with the following classes:
None

4.4.44 PlPGE Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This is the base class within a generalization heirachy that describes PGEs. The class defines abstract operations required for the planning subsystem to work out when a PGE needs to be scheduled. As well as containing the key attributes defining the PGE.

Attributes:

myInputDataTypeList - List of input data types needed by the PGE
Data Type: List
Privilege: Private
Default Value:

myInstrument - Instrument for which the PGE is appropriate
Data Type: String
Privilege: Private
Default Value:

myOutputDataTypeList - List of output data types needed by the PGE
Data Type: List
Privilege: Private
Default Value:

myPGEName - Name of the PGE
Data Type: String
Privilege: Private
Default Value:

myPGEVersion - Version number of the PGE
Data Type: float
Privilege: Private
Default Value:

myPlatform - Platform for which the PGE is appropriate, may be a list
Data Type: String
Privilege: Private
Default Value:

myTestOperational - Indicates if the PGE's status is test or operational
Data Type: enum

Privilege: Private
Default Value:

Operations:

Delete - Delete the PGE from the PDPS database
Arguments:
Return Type: void
Privilege: Public

FindDataAvailability -
Arguments: Interval
Return Type: void
Privilege: Public
PDL:{
// Iterates over the input data type list invokes the
// Find Data Availability method for each input type
}

GenerateDPRs - Generate the Data Processing Requests to fulfill the Production Request.
Arguments: PlProductionRequest
Return Type: void
Privilege: Public
This is an abstract operation

Modify
Arguments: TestOrOper:enum
Return Type: void
Privilege: Public

PlPGE - Constructor method
Arguments:
Return Type: PlPGE
Privilege: Public

PlPGE - Construct the PGE from the PDPS database
Arguments: int PGEid
Return Type: PlPGE
Privilege: Public

UpdateVersion - Update the version of the PGE
Arguments:

Return Type: void
Privilege: Public

~PlPGE - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlPGE class has associations with the following classes:
Class: PlDPR
Class: PlDataType
Class: PlProductionRequest

4.4.45 PlPGEActivity Class

Parent Class: PlActivity
Public: NoDistributed Object: No
Purpose and Description:
This class is a generalization of the PlActivity class. The class describes a Data Processing Request - a run of a PGE - within the plan.

Attributes:

All Attributes inherited from parent class

Operations:

Cancel - Cancel the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the cancel method of the Data Processing Request
}

Modify - Modify the activity
Arguments:
Return Type: void

Privilege: Public
PDL:{
// calls the modify method of the Data Processing Request
}

PlPGEActivity - Constructor method
Arguments:
Return Type: PlPGEActivity
Privilege: Public

PlPGEActivity - Constructs an activity within the plan for the Data Processing Request
Arguments: PlDPR
Return Type: PlPGEActivity
Privilege: Public

Schedule - Schedule the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the schedule method of the Data Processing Request
}

Status - Modify the activity
Arguments:
Return Type: void
Privilege: Public
PDL:{
// calls the modify method of the Data Processing Request
}

~PlPGEActivity - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlPGEActivity class has associations with the following classes:
Class: PlDPR

4.4.46 PlPGEProfile Class

Parent Class: Not Applicable

Public: YesDistributed Object: No

Purpose and Description:

This class describes the collection of information that describes a PGE to the Planning subsystem.

Attributes:

None

Operations:

$PlPGEProfile - Constructor function

Arguments: ...

Return Type: void

Privilege: Public

DeletePGEProfile - Deletes a given PGE profile

Arguments: PGEProfID:int

Return Type: void

Privilege: Public

ModifyPGEProfile - Modifies a given PGE profile

Arguments: PGEProfId:int, ModType:enum, ModField:String, ModValue:String

Return Type: void

Privilege: Public

RetrievePGEProfile - Retrieves a given PGE profile

Arguments: PlPGEProfID

Return Type: void

Privilege: Public

~PlPGEProfile - Destructor function

Arguments: ...

Return Type: void

Privilege: Public

Associations:

The PlPGEProfile class has associations with the following classes:
None

4.4.47 PlPerformance Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes the performance statistcs of a PGE. These performance statistics are established at AI&T. The class also contains attributes to describe the statistics updated from the Data Processing subsystem.

Attributes:

myElapsedTime - Elapsed time for the PGE during AI&T
Data Type: Time
Privilege: Private
Default Value:

myMaxMemoryUse - Maximum memory required by the PGE during AI&T
Data Type: float
Privilege: Private
Default Value:

myNoOfBlockInOper - Number of input blocks for the PGE during AI&T
Data Type: int
Privilege: Private
Default Value:

myNoOfBlockOutOper - Number of output blocks for the PGE during AI&T
Data Type: int
Privilege: Private
Default Value:

myNoOfPageFaults - Number of PGE page faults for the PGE during AI&T
Data Type: int
Privilege: Private
Default Value:

myNoOfSwaps - Number of PGE swaps for the PGE during AI&T
Data Type: int
Privilege: Private
Default Value:

myPGECPUTime - CPU required for the PGE during AI&T
Data Type: Time
Privilege: Private
Default Value:

myRunElapsedTime - Elapsed time for the PGE during production
Data Type: Time
Privilege: Private
Default Value:

myRunMaxMemoryUse - Maximum memory required by the PGE during production
Data Type: float
Privilege: Private
Default Value:

myRunNoOfBlockInOper - Number of input blocks for the PGE during production
Data Type: int
Privilege: Private
Default Value:

myRunNoOfBlockOutOper
Data Type: int
Privilege: Private
Default Value:

myRunNoOfPageFaults - Number of PGE page faults for the PGE during production
Data Type: int
Privilege: Private
Default Value:

myRunNoOfSwaps - Number of PGE swaps for the PGE during production
Data Type: int
Privilege: Private
Default Value:

myRunPGUCPUTime - CPU required for the PGE during production
Data Type: Time
Privilege: Private
Default Value:

myRunSharedMemoryUse - Shared memory required by the PGE during production
Data Type: float
Privilege: Private
Default Value:

mySharedMemoryUse - Shared memory required by the PGE during AI&T
Data Type: float
Privilege: Private
Default Value:

Operations:

UpdateRunTimePerfPar
Arguments: ParId:int, ParValue:String
Return Type: Void
Privilege: Public

Associations:

The PlPerformance class has associations with the following classes:
PlPGEProfile (Aggregation)

4.4.48 PlPlan Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.

Attributes:

myActiveStatus - Indicator to identify active plan
Data Type: Boolean
Privilege: Private
Default Value: False

myDescription - Descriptive text for operator comments describing conditions under which the plan was generated.
Data Type: String
Privilege: Private
Default Value:

myEndTime - End time for the plan
Data Type: Time
Privilege: Private
Default Value:

myPlanName - Descriptive name for plan to facilitate plan selection.
Data Type: String
Privilege: Private
Default Value:

myStartTime - Start time for the plan
Data Type: Time
Privilege: Private
Default Value:

Operations:

CreatePlan - Create an entry in the PDPS database describing the Plan
Arguments:
Return Type: void
Privilege: Public

DeletePlan - Deletes the plan from the PDPS database along with the associated activities.
Arguments:
Return Type: void
Privilege: Public

PlanProductionRequest - Adds the Data Processing Requests associated to a Production Plan. This involves creating an activity within the plan to describe the DPR, and allocating resources for the DPR.
Arguments: PlProductionRequest: PR
Return Type: void
Privilege: Public

PlanSchedule - This operation is an abstraction that describes the planning process by which activities are associated to resources in order to set out a time-line for the plan.
Arguments:

Return Type: void
Privilege: Public

Publish - Initiates the activities involved in publishing a plan.
Arguments:
Return Type: void
Privilege: Public

UnplanProductionRequest - Remove the activities associated with a particular production request from the plan.
Arguments: PlProductionRequest: PR
Return Type: void
Privilege: Public

UpdatePlan - Updates the plan from the status of the activities in the PDPS database.
Arguments:
Return Type: void
Privilege: Public
PDL:{
// Create an ordered list of the Ground Events that are in the plan

// Verify whether any new ground events have been added
// If so alert operator, with option of including event

// Regenerate the plan
}

Associations:

The PlPlan class has associations with the following classes:
Class: PlActivity
Class: PlProductionPlannersUI

4.4.49 PlPlanASCIIReportFile Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class represents an ascii readable externalized report format of the plan. The precise format of this report is TBD. The report will provide a summary view of the tasks planned (no. of particular product types generated within the plan period) as well as a prediction of

the generation time for each product.

Attributes:

None

Operations:

Write
Arguments:

Associations:

The PlPlanASCIIReportFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

4.4.50 PlPlanBinaryReportFile Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class represents an electronic parsable report format of the plan. The precise format of this report is TBD. The report will provide a summary view of the tasks planned (no. of particular product types generated within the plan period) as well as a prediction of the generation time for each product.

Attributes:

None

Operations:

Write
Arguments:

Associations:

The PlPlanBinaryReportFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

4.4.51PlPlanGenerationUI Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The class shows the functions available to the production scheduler in support of generating a plan

Attributes:

None

Operations:

AddPRtoPlan - Add the Data Processing Requests associated to a Production Requests to a plan.
Arguments: PlProductionRequest: PR
Return Type: void
Privilege: Public

DelPRfromPlan - Delete the Data Production Requests associated to a Production Request from a plan.
Arguments: PlProductionRequest: PR
Return Type: void
Privilege: Public

NewPlan - Generate a new plan
Arguments:
Return Type: void
Privilege: Public

SelectPlan - Select a plan for modification or activation
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlPlanGenerationUI class has associations with the following classes:
Class: PlProductionRequest
PlPlanningWorkbenchUI (Aggregation)

4.4.52 PlPlanMetadataFile Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class represents the metadata that will be used to describe the plan within the Document Data Server. The format of this report will be a Parameter=Value list line separated.

Attributes:

myDAAC - The site at which the plan was produced
Data Type: String
Privilege: Private
Default Value:

myDataSets - Describes the data sets which are planned for production within the plan.
Data Type: String
Privilege: Private
Default Value:

myDescription - Captures the operators description of the purpose of the plan (for example 30 day forecast)
Data Type: String
Privilege: Private
Default Value:

myEndDay - The end date of the plan
Data Type: Time
Privilege: Private
Default Value:

myForecast - The number if days time period which the plan covers
Data Type: int
Privilege: Private
Default Value:

myStartTime - The start date of the plan
Data Type: Time
Privilege: Private
Default Value:

Operations:

None

Associations:

The PlPlanMetadataFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

4.4.53 PlPlanningWorkbenchUI Class

Parent Class: PlApplication
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool.

Attributes:

All Attributes inherited from parent class

Operations:

ProductionSchedulerCommands
Arguments:
Return Type: Void
Privilege: Public

ResourceManagerCommands
Arguments:
Return Type: Void
Privilege: Public

Associations:

The PlPlanningWorkbenchUI class has associations with the following classes:
Class: PlPlan
Class: PlPublishScript

4.4.54 PlProductionPlannersUI Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool. The class does describe the basic operations that are provided from the interface.

Attributes:

PlanSelectionWindow
Data Type:
Privilege: Private
Default Value:

SchedulingPeriod
Data Type:
Privilege: Private
Default Value:

Operations:

ActivateSchedule
Arguments:
Return Type: Void
Privilege: Public

AddPRtoPlan
Arguments:
Return Type: Void
Privilege: Public

CancelSchedule
Arguments:
Return Type: Void
Privilege: Public

DelPRfromPlan
Arguments:
Return Type: Void
Privilege: Public

DeletePlan
Arguments:
Return Type: Void
Privilege: Public

ModifySchedule
Arguments:
Return Type: Void
Privilege: Public

NewPlan
Arguments:
Return Type: Void
Privilege: Public

SelectPlan
Arguments:
Return Type: Void
Privilege: Public

StatusSchedule
Arguments:
Return Type: Void
Privilege: Public

Associations:

The PlProductionPlannersUI class has associations with the following classes:
Class: PlPlan

4.4.55 PlProductionRequest Class

Parent Class: Not Applicable

Public: YesDistributed Object: No

Persistent Class: True

Purpose and Description:

A production request describes an order for data sets, a production request typically specifies a request for a Data Set to be produced for an extended period of time (e.g a month's worth of some product).

Attributes:

myDataCollectionStartTime - Start time for the production request.

Data Type: Time

Privilege: Private

Default Value:

myDataCollectionStopTime - Stop time for the production request.

Data Type: Time

Privilege: Private

Default Value:

myOutputDataType - Identifies the product desired by the ECS User.

Data Type: String

Privilege: Private

Default Value:

myPGEIdentifier - Identifies the PGE

Data Type: String

Privilege: Private

Default Value:

myPriority - User requested priority for the submitted production request.

Data Type: int

Privilege: Private

Default Value:

Operations:

DefinePGERuns - Defines the production requests corresponding to the production request.

Arguments:

Return Type: Void

Privilege: Public

Modify - Modifies the production request.
Arguments: ModField:String, ModVal:String
Return Type: Void
Privilege: Public

PlProductionRequest - Constructor for the PlProductionRequest class.
Arguments: Product:String, PGE:String, Start:Time, Stop:Time, Prior:int
Return Type: Void
Privilege: Public

RetrieveAllProdReq - Retrieves all the production requests currently in effect
Arguments:
Return Type: ProductionRequest *
Privilege: Public

Associations:

The PlProductionRequest class has associations with the following classes:
Class: PlDPR
Class: PlPGE
Class: PlProductionRequestUI

4.4.56 PlProductionRequestUI Class

Parent Class: PlApplication
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the production request editor application.

Attributes:

All Attributes inherited from parent class

Operations:

AddProductionRequest
Arguments:

Return Type: void
Privilege: Public

DelProductionRequest
Arguments:
Return Type: void
Privilege: Public

ModifyProductionRequest
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlProductionRequestUI class has associations with the following classes:
Class: PlDataTypeCatalogue
Class: PlProductionRequest

4.4.57 PlPublishScript Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class depicts the script required to send a message to a preprepared list of users when a new plan is published.

Attributes:

None

Operations:

Execute
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlPublishScript class has associations with the following classes:
Class: PlAddressFile
Class: PlPlanningWorkbenchUI

4.4.58 PlPublishedPlan Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This object encapsulates the methods requred to insert externalized formats of the plan into the document data server

Attributes:

None

Operations:

InsertInDDS - Insert the published plan into the Document Data Server
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlPublishedPlan class has associations with the following classes:
Class: DsCtClient
Class: DsCtCommand
Class: PlPlan

4.4.59 PlResource Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:

This class is the base class in a generalization heirachy describing the production resource.

Attributes:

myID - Unique identifier for the resource
Data Type: int
Privilege: Private
Default Value:

myName - The name of the resource
Data Type: String
Privilege: Private
Default Value:

Operations:

None

Associations:

The PlResource class has associations with the following classes:
PlResourceConfigeration (Aggregation)

4.4.60 PlResourceConfigeration Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class describes the resource configuration of the Production Resources.

Attributes:

None

Operations:

BuildConfiguration - Build the resource configuration from the MSS services.
Arguments:
Return Type: void

Privilege: Public

Associations:

The PlResourceConfigeration class has associations with the following classes:
Class: MsDAAC
Class: PlResourceManager

4.4.61 PlResourceManager Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class represents an abstraction for the resource management capabilities used when generating a plan, describing the operations required to match resource requirements of an activity to the available resources, and to allocate the resource for the activity.

Attributes:

None

Operations:

AllocateResources - Allocate the resources to an activity
Arguments: PlResource: Resource, PlActivity: Activity
Return Type: Void
Privilege: Public

DeallocateResources - Deallocate the resources for an activity
Arguments: PlResource: Resource, PlActivity: Activity
Return Type: Void
Privilege: Public

MatchResourceRequirement - Match the resource requirements of an activity to a resource
Arguments: PlResourceRequirements
Return Type: PlResource
Privilege: Public

Associations:

The PlResourceManager class has associations with the following classes:
Class: PlPlan

4.4.62 PlResourceManagersUI Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The class presents the functions that are available to the resource manager role.

Attributes:

EventDescriptionWindow
Data Type:
Privilege: Private
Default Value:

EventSelectionWindow
Data Type:
Privilege: Private
Default Value:

Operations:

ConfirmCreate
Arguments:
Return Type: void
Privilege: Public

ConfirmDelete
Arguments:
Return Type: void
Privilege: Public

ConfirmModify
Arguments:
Return Type: void

Privilege: Public

CreateGroundEvent
Arguments:
Return Type: void
Privilege: Public

DeleteGroundEvent
Arguments:
Return Type: void
Privilege: Public

ModifyGroundEvent
Arguments:
Return Type: void
Privilege: Public

SelectGroundEvent
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlResourceManagersUI class has associations with the following classes:
Class: PlGroundEvent
PlPlanningWorkbenchUI (Aggregation)

4.4.63PlResourceRequirement Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class contains a description of the resource requirements of a PGE, which may be matched against the resource configuration known to the Planning subsystem.

Attributes:

myComputer - A computer within the string required by the PGE
Data Type: String
Privilege: Private
Default Value:

myDiskSpace - The disk space required for a PGE
Data Type: int
Privilege: Private
Default Value:

myNCPUs - The number of CPUs required for a PGE
Data Type: int
Privilege: Private
Default Value:

myOperatingSystem - The operating system for which a PGE is configured
Data Type: String
Privilege: Private
Default Value:

myString - The string required by the PGE
Data Type: String
Privilege: Private
Default Value:

Operations:

PlResourceRequirement - Constructor method
Arguments:
Return Type: PlResourceRequirement
Privilege: Public

~PlResourceRequirement - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlResourceRequirement class has associations with the following classes:
Class: PlDPR
Class: PlGroundEvent

4.4.64 PlRoutineArrival Class

Parent Class: PlDataSource
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
This class is a specialization of the PlDataSource class and describes the most frequent method for predicting data arrivals within the ECS (at least for the TRMM data sets). This class contains the attributes and operations required to describe routine ingest of external data.

Attributes:

myDataBoundary - Identifies the time boundary for each Data Granule.
Data Type:
Privilege: Private
Default Value:

myDataPeriod - Identifies the data collection period within each Data Granule.
Data Type: Time
Privilege: Private
Default Value:

myDelay - The average delay between data collection and the arrival that of the Data Granule within ECS.
Data Type: Time
Privilege: Private
Default Value:

Operations:

PredictArrivals - Predicts data instance arrivals for a given period.
Arguments: Start:Time, Stop:Time
Return Type: void

Privilege: Public

Associations:

The PlRoutineArrival class has associations with the following classes:
None

4.4.65 PlScheduledArrival Class

Parent Class: PlDataSource
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class implements another method that predicts data granule arrivals based on an external schedule from another site. This specialization is not required for the TRMM release.

Attributes:

All Attributes inherited from parent class

Operations:

PredictArrivals
Arguments: Start:Time, Stop:Time
Return Type: void
Privilege: Public

Associations:

The PlScheduledArrival class has associations with the following classes:
None

4.4.66 PlString Class

Parent Class: PlResource
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
A string describes the logical collection of a number of resources allocated for an instruments processing needs

Attributes:

myComputerList - Describes the list of computers that make up the string
Data Type: List
Privilege: Private
Default Value:

Operations:

AddComputer - Add a computer to the list
Arguments: PlComputer: Comp
Return Type: void
Privilege: Public

RemoveComputer - Remove a computer from the list
Arguments: PlComputer: Comp
Return Type: void
Privilege: Public

Associations:

The PlString class has associations with the following classes:
Class: PlComputer

4.4.67 PlSubscriptionManager Class

Parent Class: PlApplication
Public: NoDistributed Object: No
Purpose and Description:
This class contains the main application methods associated with the subscription manager

application

Attributes:

All Attributes inherited from parent class

Operations:

DeinstallReceivers - Deinstalls the receivers for each data type that is subscribed to
Arguments:
Return Type: void
Privilege: Public

Init - Initializes the application by creating the lifecycle service callbacks (shutdown etc.) and setting up the communications for receiving subscriptions
Arguments:
Return Type: void
Privilege: Public

InstallReceivers - Installs the receivers for each data type that is subscribed to
Arguments:
Return Type: void
Privilege: Public

Shutdown - Performs a graceful shutdown of the application by deinstalling receivers and closing communications
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlSubscriptionManager class has associations with the following classes:
Class: EcMpMsgPsngCtrl
Class: MsManager isaserver
Class: MsEvent logs
Class: PlDataTypes registersnotificationof

4.4.68 PlSubscriptionManagerCallbacks Class

Parent Class: MsMgCallbacks

Public: NoDistributed Object: No

Purpose and Description:

Specialization class to provide the subscription manager callbacks to the MsManager in order to be activated for the appropriate lifecycle services (shutdown, resume, suspend notify).

Attributes:

All Attributes inherited from parent class

Operations:

NotifyCb - Notify callback not used at Release A. A null method will be provided.

Arguments:

Return Type: void

Privilege: Public

ResumeCb - Resume not supported at Release A.

Arguments:

Return Type: void

Privilege: Public

ShutdownCb - Shutdown invokes the subscription manager shutdown method, deinstalling the receiving queues from the subscription notification queues and terminating the program in a clean manner.

Arguments:

Return Type: void

Privilege: Public

SuspendCb - Suspend not supported at Release A

Arguments:

Return Type: void

Privilege: Public

Associations:

The PlSubscriptionManagerCallbacks class has associations with the following classes:

None

4.4.69 PlSubscriptionSubmitIF Class

Parent Class: PlApplication
Public: NoDistributed Object: No
Purpose and Description:
This class is an abstraction for the user interface to the subscription submission application. The interface will be developed with a suitable GUI building tool.

Attributes:

DataTypeSelectionWindow
Data Type:
Privilege: Private
Default Value:

SubscriptionSubmissionControl
Data Type:
Privilege: Private
Default Value:

Operations:

DisplayDataTypes - This operation displays a list of the dynamic data types known to the Planning subsystem as required input to a PGE.
Arguments:
Return Type: Void
Privilege: Public

Initialize - Initialize the subscription submission GUI.
Arguments:
Return Type: Void
Privilege: Public

SelectDataType - This operation allows the user to select a Data Type from the displayed list.
Arguments:
Return Type: Void
Privilege: Public

SubmitSubscription - Submit a subscription for the selected Data Type.
Arguments:
Return Type: Void
Privilege: Public

WithdrawSubscription - Withdraw the subscription for the selected Data Type.
Arguments:
Return Type: Void
Privilege: Public

Associations:

The PlSubscriptionSubmitIF class has associations with the following classes:
Class: IoAdAdvertisingSrv_C
Class: DsClSubscription creates
Class: IoAdServiceCollection_C searches
Class: IoAdServiceAdvertisement selects
Class: PlDataType updates

4.4.70 PlTimeLineDisplay Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Purpose and Description:
This class describes the user interface component that represents the graphical display of a plan. This will be implemented by a COTS or re-use component.

Attributes:

None

Operations:

DisplayPlan - Display the specified Plan.
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlTimeLineDisplay class has associations with the following classes:
PlPlanningWorkbenchUI (Aggregation)

4.4.71 PlTimeScheduled Class

Parent Class: PlPGE
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc.

Attributes:

myAcquisitionProcessingBoundary - Identifies the time needed (for data) for extrapolation purposes.
Data Type: Time
Privilege: Private
Default Value:

myAcquisitionProcessingPeriod - Identifies the acquisition period for the input data.
Data Type: Time
Privilege: Private
Default Value:

myNoOfTimeUnitsPerPgeRun - Number of Units of time for PGE run.
Data Type: float
Privilege: Private
Default Value:

myTimeUnits - Units of time for PGE run frequency.
Data Type: enum
Privilege: Private
Default Value:

Operations:

GenerateDPRs - { // Iterate over the period of the production request in steps // of myAquisitionProcessingPeriod from the nearest occurance // of the myAcquisitionProcessingBoundary within the request period // Generate a DPR // Invoke the select data method for the DPR }
Arguments: PlProductionRequest
Return Type: void
Privilege: Public

Associations:

The PlTimeScheduled class has associations with the following classes:
None

4.4.72 PlUser Class

Parent Class: Not Applicable
Public: NoDistributed Object: No
Persistent Class: True
Purpose and Description:
This class describes the privilidges associated with a user.

Attributes:

myProdReqPriv - Indicates whether the user has privilidges to enter / modify production requests.
Data Type: Boolean
Privilege: Private
Default Value:

myProdSchPriv - Indicates whether the user has privilidges to create plans, and to schedule production
Data Type: Boolean
Privilege: Private
Default Value:

myRescManPriv - Indicates whether the user has privilidges to enter resource ground events.
Data Type: Boolean

Privilege: Private
Default Value:

myUserName - The ECS user name for the user.
Data Type: String
Privilege: Private
Default Value:

Operations:

Authenticate - Authenticates the user in order to run the planning subsystem software.
Arguments:
Return Type: Boolean
Privilege: Public

Associations:

The PlUser class has associations with the following classes:
Class: PlApplication

4.4.73 PlUserParameters Class

Parent Class: Not Applicable
Public: YesDistributed Object: No
Persistent Class: True
Purpose and Description:
Describes any user defined parameters that are associated to a PGE.

Attributes:

myDefaultValue - The default value for the user parameter.
Data Type: String
Privilege: Private
Default Value:

myDescription - Describes the user parameter
Data Type: String
Privilege: Private
Default Value:

myLogicalID - Id of the user parameter
Data Type: int
Privilege: Private
Default Value:

myName - Name of the user parameter
Data Type: String
Privilege: Private
Default Value:

Operations:

Create - Create an entry within the PDPS database
Arguments:
Return Type: void
Privilege: Public

Delete - Delete the entry from the PDPS database
Arguments:
Return Type: void
Privilege: Public

Modify - Modifies the default value of the user parameter
Arguments: DefValue:String
Return Type: void
Privilege: Public

PlUserParameters - Constructor method
Arguments:
Return Type: PlUserParameters
Privilege: Public

~PlUserParameters - Destructor method
Arguments:
Return Type: void
Privilege: Public

Associations:

The PlUserParameters class has associations with the following classes:
Class: PlPGE

PlPGEProfile (Aggregation)

4.5 PLANG Dynamic Model

The PLANG Dynamic model presents a number of scenarios and event traces that describe the key interactions of the classes participating in the various components of the Planning subsystem.

4.5.1 PGE Profile

This scenario describes various operations related to PGE profiles. PGE profiles are captured during Algorithm Integration and Testing (AI&T). PGE profiles describe a PGE itself (e.g., executable, scripts), user parameters, input and output data types, PGE resource requirements, and PGE performance statistics.

PGE information includes the periodicity of the PGE - i.e., how often the PGE can be run. Certain PGEs can be run only when an input data type is scheduled to be acquired. Certain PGEs will be run at regular intervals (for example, once a day, or once a month). More details are indicated in the object model (Figure 4.3-1).

User parameter information includes default user parameters. The ECS User can override these default values with any other values from Production Requests.

Input and output data types are described by a single data type. An output data type from a PGE can be the input data type for another PGE. This situation is referred to as data chaining. Data types can be static in nature (e.g., calibration data) or dynamic (e.g., L0 data, ancillary data). A non-L0 input data type needed by a PGE may be generated within the DAAC, or it may be generated at another ECS site.

PGE resource requirements information includes identification of hardware types and any particular hardware items associated with the PGE.

PGE performance refers to PGE performance information (e.g., CPU time). Some of this information is gathered during Algorithm and Integration Test (AI&T), and the rest are gathered in the operational environment when the PGE is run.

4.5.1.1 Beginning Assumptions

None

4.5.1.2 Interfaces With Other Subsystems and Segments

PGE profiles are entered/modified/deleted using the PGE Profile Editor (provided by Algorithm Integration and Test Tools (AITTL) CSCI in Data Processing subsystem). PGE Profile Editor utilizes PlPGEProfile class to enter and manipulate PGE profile in the system.

4.5.1.3 Stimulus

PGE profiles are entered/modified/deleted using the PGE Profile Editor (provided by AITTL CSCI in Data Processing subsystem).

4.5.1.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

DpPGEProfileEditor (AI&T CSCI in Data Processing subsystem)

PlPGEProfile

PlResourceRequirement

PlUserParameters

PlPGE

PlDataType

PlDataTypeRequirement

4.5.1.5 Beginning System, Segment and Subsystem State(s)

PDPS Database is in a steady state, up and running.

4.5.1.6 Ending State

PDPS Database is in steady state, up and running.

4.5.1.7 Scenario Description

Thread 1: PGE profile entry

1. A PGE profile is entered using the PGE Profile Editor. As described before, PGE profile includes PGE itself (e.g., executables, scripts, combination of executables and scripts), user parameters, input and output data types, and PGE resource requirements.
2. PlPGEProfile class creates corresponding PlResourceRequirements class, PlUserParameters class, PlPge class, PlDataType classes for all input and output data types. The PGE version number would be 1.0. The status of the PGE is set by default to Test.

Thread 2: PGE profile modification

1. The User expresses the wish to modify a PGE profile through the PGE Profile Editor.
2. DpPgeProfileEditor class in AITTL CSCI in Data Processing subsystem sends a request to PlPGEProfile class for a list of existing PGEs and their version numbers.
3. PlPgeProfile class sends the list of existing PGEs and their version numbers along with brief descriptions to DpPgeProfileEditor class. DpPgeProfileEditor class displays the list.
4. User selects the PGE profile and version number that needs to be modified. If the version number is not selected, the default is the latest version. DpPgeProfileEditor class sends PGE profile modification command along with the PGE profile and version details to PlPgeProfile class.
5. PlPgeProfile class sends the details regarding the requested PGE profile to DpPgeProfileEditor class, which is displayed to the user.

Sub-Thread 2-1: Value(s) associated with a PGE is modified.

6. The User modifies the desired field(s). The fields that are modified will determine if the modification(s) are major or minor in nature. Let us assume the version number of the existing PGE (that is being modified) is X.Y. If the modifications is major, the new version will be numbered X+1.0. If the modifications is minor, the new version will be numbered X.Y+1.

An example of a minor modification is change in the default value of a user parameter. An example of a major modification is input data type is changed, or the PGE is changed.

Sub-Thread 2-2: A PGE is promoted/demoted between Test and Operation.

6. The User changes the status of the PGE from Test to Operational.

Thread 3: PGE profile deletion

A PGE profile is deleted when the User wants it to be completely purged from the system.

1. The User expresses the wish to delete a PGE from the PGE Profile Editor.
2. DpPgeProfileEditor class in AITTL CSCI in Data Processing subsystem sends a request to PlPGEProfile class for a list of existing PGEs and their version numbers.
3. PlPgeProfile class sends the list of existing PGEs and their version numbers along with brief descriptions to DpPgeProfileEditor class. DpPgeProfileEditor class displays the list..
4. User selects the PGE profile version that needs to be deleted. DpPgeProfileEditor class deletes corresponding PlResourceRequirements class, PlUserParameters class, PlPge class, PlDataType classes for all input and output data types.

4.5.1.8 Event Trace

See Figures 4.5-1 to 4.5-3.

4.5.2 Production Requests

This scenario describes various operations related to entering production requests. The production request is entered by the Production Scheduler. The Production Scheduler has the option of entering a particular PGE to be used during production. If no PGE is specified Planning will determine the default PGE required. This scenario describes the process by which a production request is translated to multiple data processing requests.

When a standard production request is entered, the input data granules needed to fulfil the request may already be available to ECS, or they may be available in the future. Availability of input data depends on data collection times entered in the production request with respect to when the request is entered. The procedure by which the availability of data is determined is described in scenario 4.5.3.

4.5.2.1 Beginning Assumptions

None

4.5.2.2 Interfaces With Other Subsystems and Segments

Standard Production Requests are entered / modified / deleted by the Production Scheduler (human). These requests are handled by PlProductionRequestUI class. Necessary GUI will be developed to enter these requests.

4.5.2.3 Stimulus

Standard Production request is entered / modified / deleted by the Production Scheduler.

4.5.2.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

PlPlanningUserUI

PlProductionRequest

PlDataTypeCatalogue

PlDataType

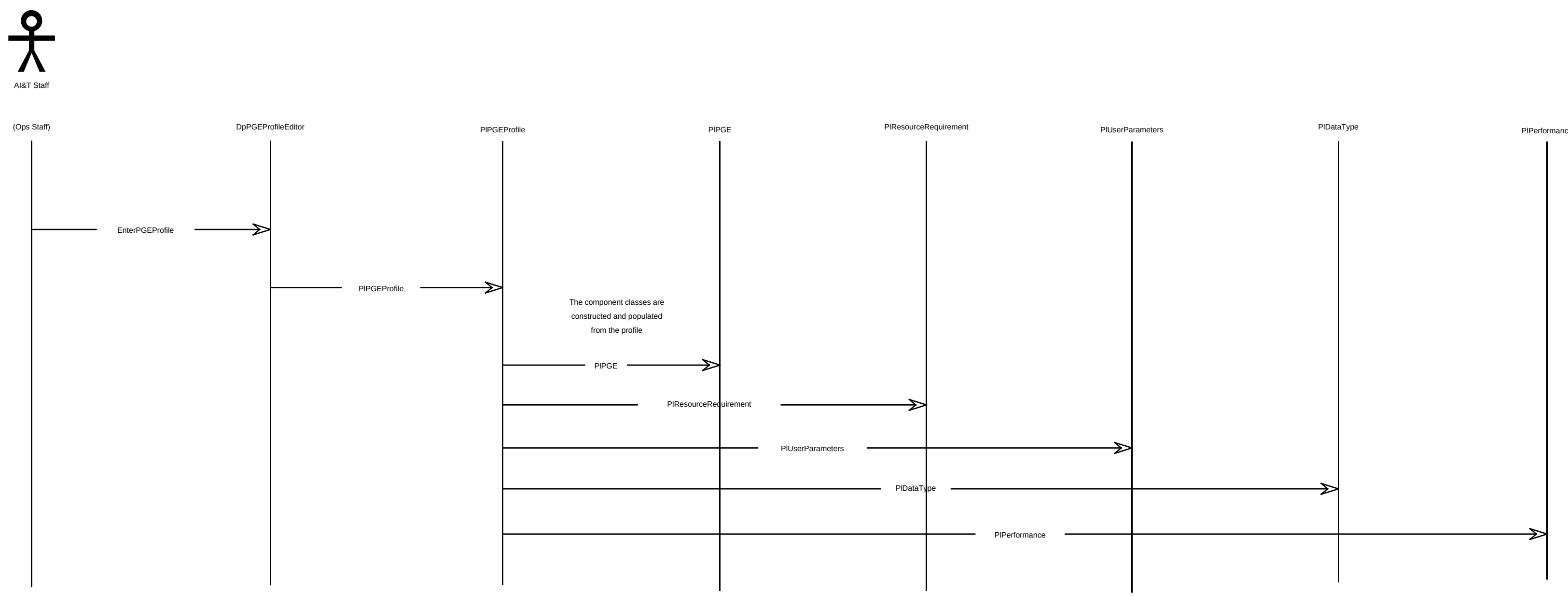


Figure 4.5-1. PGE Profile Entry Event Trace

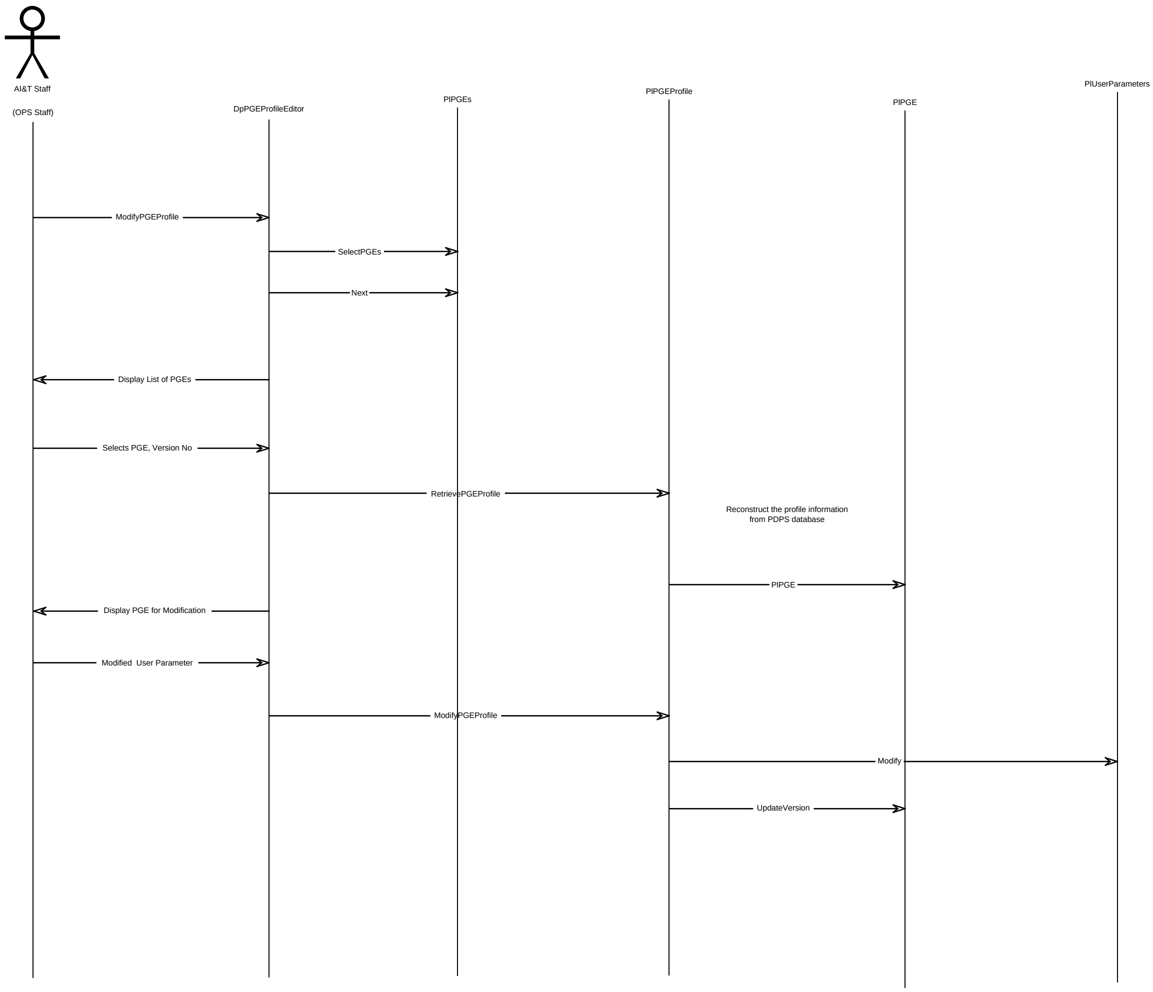


Figure 4.5-2. PGE Profile Modification Event Trace

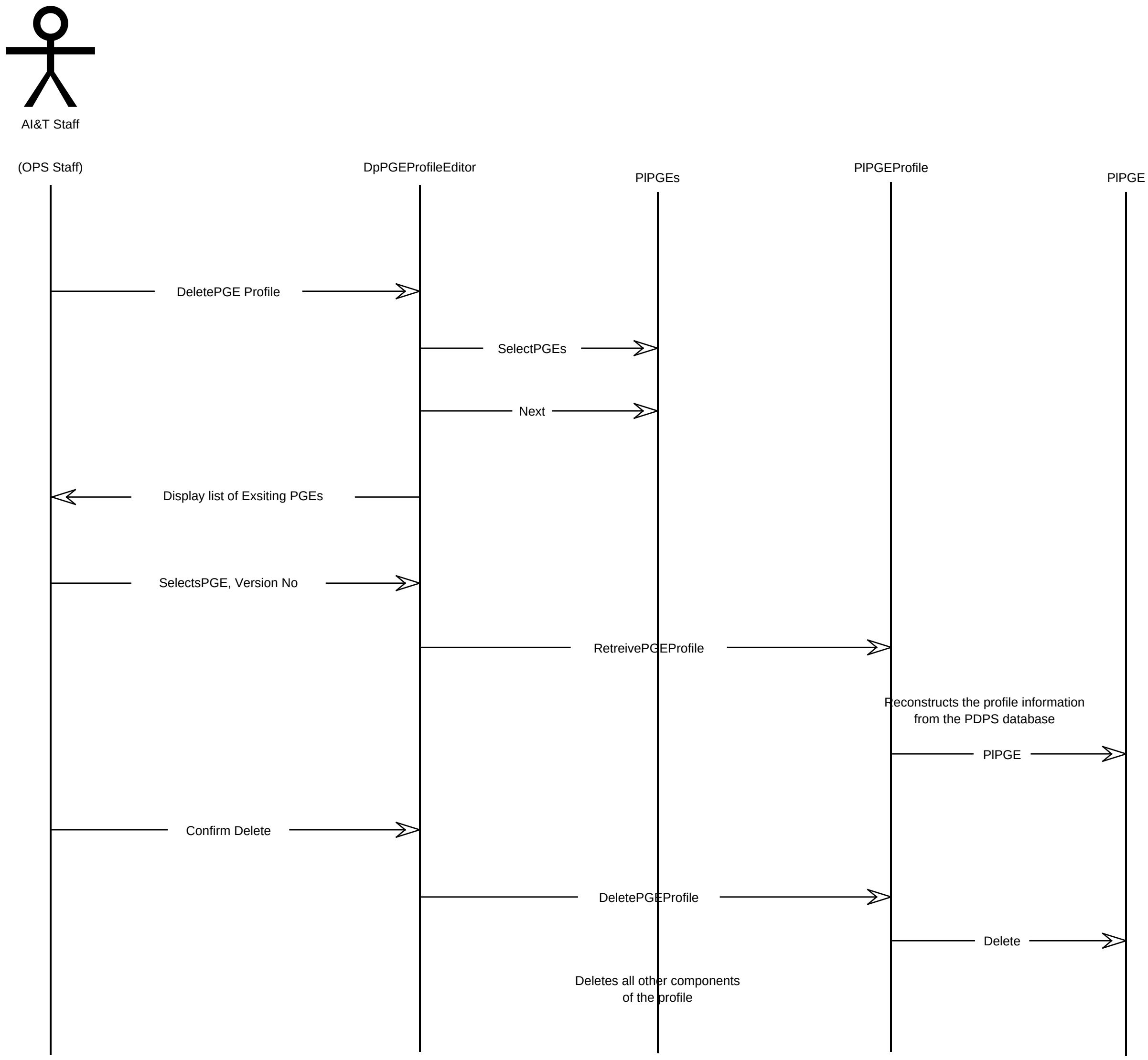


Figure 4.5-3. PGE Profile Deletion Event Trace

PlPGE

PlDPR

PlDataTypeRqt

PlInputDataType

PlDataGranule

4.5.2.5 Beginning System, Segment and Subsystem State(s)

PDPS Database is in a steady state, up and running.

4.5.2.6 Ending State

PDPS Database is in steady state, up and running.

4.5.2.7 Scenario Description

Thread 1: Standard Production request entry

1. Using the Production Request Editor, the Production Scheduler will enter the command for initiating the entry of a Standard Production request. This command is received by PlProductionRequestUI class.
2. PlProductionRequestUI class will send a request to PlDataTypeCatalogue class for a list of products that can be generated (product catalogue). For each product corresponding PGE and input data types are also requested.
3. PlDataTypeCatalogue class will obtain the required information (through PDPS database queries), and send to PlProductionRequestUI class.
4. PlProductionRequestUI class will display the product catalogue.
5. The Production Scheduler will make the desired selection of the product. At that time, the default PGE and any default user parameters for that PGE are displayed. The Production Scheduler enters the standard production request along with (1)specific user parameters, or (2) any optional PGE and user parameters associated with that PGE. This request will also have the data collection start and end periods for product generation. Typically the data collection start and end periods refer to one of the input data types for that PGE, and this input type is represented by the PlDataType class.
6. The entered command is checked for validity. For example, data collection start time should be ahead of data collection end time (If not, the standard production request will be rejected with a warning).
7. The valid command is sent to PlProductionRequest class.
8. PlProductionRequest class sends a message to PlPGE class to generate instances of PlDataGranule class and PlDPR class.
9. PlPGE class generates PlDataGranule objects. Note: This is described in scenario 4.5.3.
10. PlPGE class has information on how the input data instances map to Data Processing Requests. Note: PlPGE class has specialization classes - PlDataScheduled, PlTimeScheduled, etc. which describe how often the given PGE needs to be run (refer to Figure 5, PGE Profile Object Model for details). PGE creates many instances of PlDPR objects based on this information.

Thread 2: Standard Production request modification

1. Using the Production Request Editor, the Production Scheduler will enter the command for initiating a Standard Production request modification. This command is received by PlProductionRequestUI class.
2. PlProductionRequestUI class will send a request to PlProductionRequest class to send a list of all current Standard Production requests.

3. PlProductionRequest class will gather the requested list (through Planning Database queries), and send the list to PlProductionRequestUI class. PlProductionRequestUI class will display the list to the Production Scheduler.
4. The Production Scheduler will modify the intended field on the intended production request (which can be data collection start and end times, or PGEs).
5. The entered command is checked for validity. For example, data collection start time should be ahead of data collection end time (If not, the standard production request modification will be rejected with a warning).
6. The valid command is sent to PlProductionRequest class.

Note: PlProductionRequest class has the pointers to PlDataGranule and PlDPR objects corresponding to the Production Request.

Sub-Thread 2-1: Data collection start/end time/both is/are modified

7. PlProductionRequest class compares original data collection period with modified data collection period.

   Any new data collection period is treated as if a new standard production request is entered (refer to Thread 1: Standard Production request entry for this case).

Sub-Thread 2-2: PGE is modified

7. When a PGE is modified, the PlPGE object corresponding to the new PGE is determined. With the new PGE, it is possible that some of the input data types may change.

   If there is no change in any of the input data types, PlDPR objects corresponding to the production request are modified to indicate the new PlPGE class.

   If there is change in any of the input data types, PlDataGranule corresponding to the old input data type of the old production request are deleted. New input data types are treated as if a new standard production request is entered (refer to Thread 1: Standard Production request entry for this case), which will result in new instances of PlDataGranule corresponding to the new input data type of the modified production request. PlDPR objects corresponding to the production request are modified to indicate the new PlPGE class, and new PlDataGranule classes.

Thread 3: Standard Production request deletion

1. Using the Production Request Editor (Standard Production request deletion window), the Production Scheduler will enter the command for initiating a Standard Production request deletion. This command is received by PlProductionRequestUI class.
2. PlProductionRequestUI class will send a request to PlProductionRequest class to send a list of all current Standard Production requests.
3. PlProductionRequest class will gather the requested list (through Planning Database queries), and send the list to PlProductionRequestUI class. PlProductionRequestUI class will display the list to the Production Scheduler.
4. The Production Scheduler will delete the intended production request.
5. The deletion command is sent to PlProductionRequest class.

6. PlProductionRequest class deletes PlDataGranule and PlDPR objects corresponding to the production request as long as the PlDataGranule and PlDPR objects correspond to an another production request, in which case they are not deleted.
7. PlProductionRequest object is deleted.

4.5.2.8 Event Trace

See Figures 4.5-4 and 4.5-5.

4.5.3 Data Availability Scenario

4.5.3.1 Abstract

This scenario describes the methods by which the data available for a Production Request are determined. Data may be already available within the Data Server (in the case of reprocessing or processing of historic data), otherwise a prediction of the arriving data is generated. This scenario describes a sequence of events that occur in the explosion of a Production Request into Data Processing Requests, as described in Scenario 4.5.2

4.5.3.2 Interfaces With Other Subsystems and Segments

The Science Data Server is queried to determine the availability of data.

4.5.3.3 Stimulus

This scenario describes the procedures by which the data available for a Production Request are determined. The overall stimulus is the receipt of a new production request. This is described in scenario 4.5.2.

4.5.3.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlProductionRequest
- PlPGE
- PlDataType
- DsClQuery
- DsClESDTReferenceCollector
- DsClESDTReference
- PlDataSource

4.5.3.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.3.6 Ending State

No change in PDPS database server state.

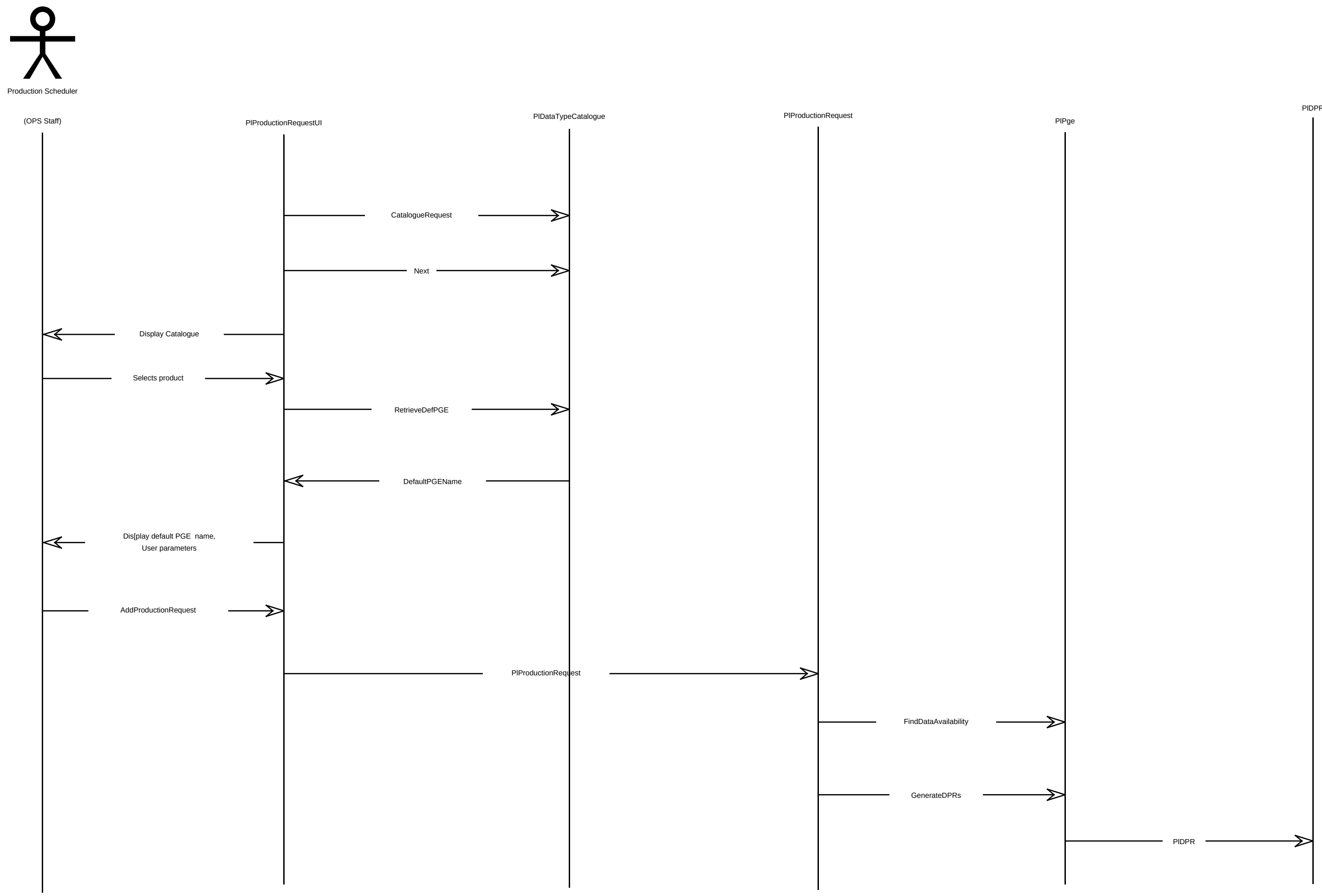


Figure 4.5-4. Production Request Entry Event Trace

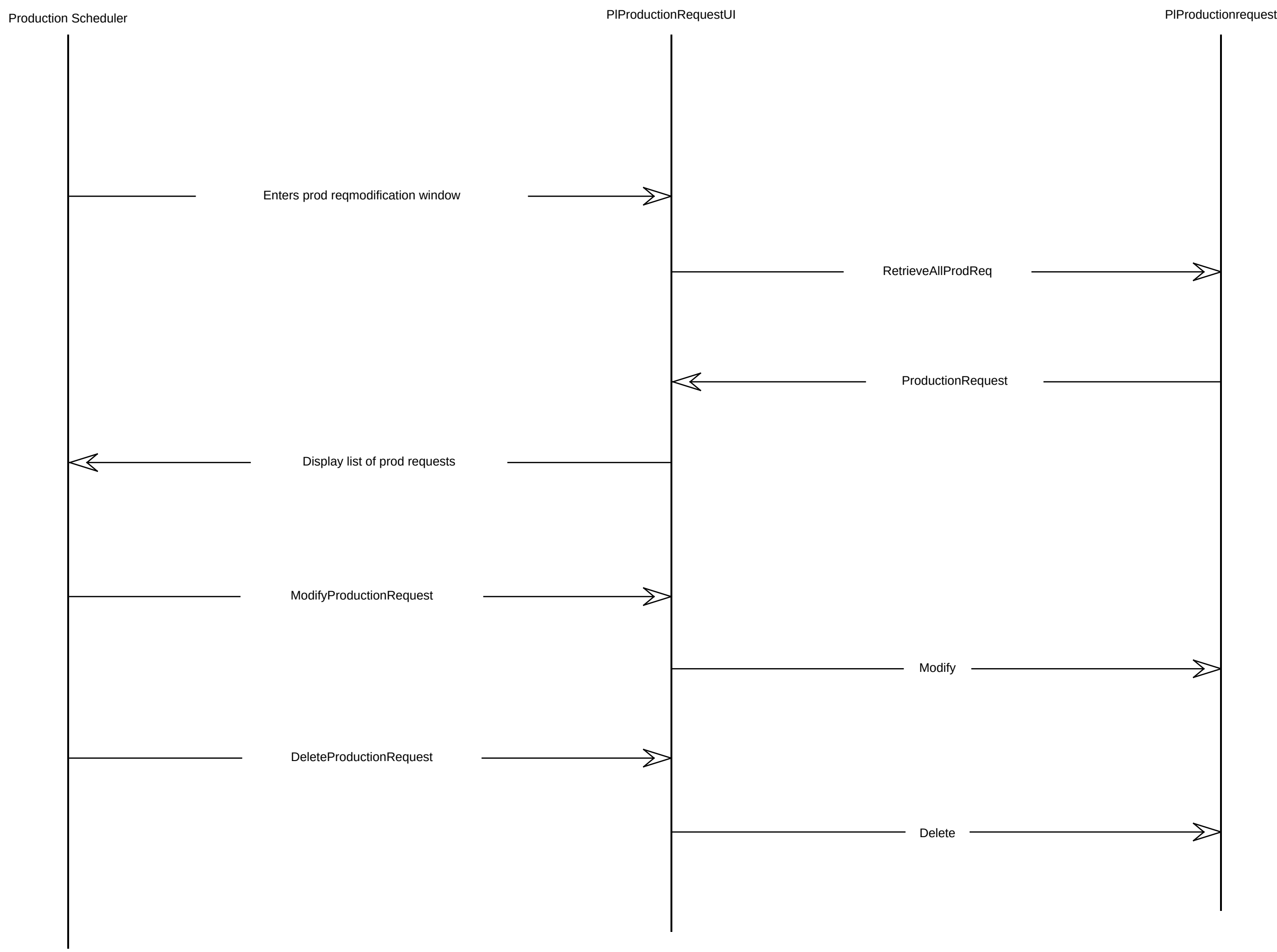


Figure 4.5-5. Production Request Deletion Event Trace

4.5.3.7 Scenario Description

1. The PlProductionRequest class invokes the FindDataAvailability method of the PGE in order to determine the availability of the data required for that production request, using the start and stop time of the Production Request
2. The PGE iterates through each of the input data types for that PGE, and invokes the FindDataAvailability for that data type.
3. Within the FindDataAvailability method it is determined whether any of the period of interest is in the past, if so the Data Server which provides for the archive of that Data Type is searched for existence of the Data.
4. A query is constructed using the DsClQuery class, which is then applied against the Data Server (as defined within the DsClESDTReferenceCollector class).
5. The results set returned from the query is iterated through, extracting the required metadata that describes the instance, creating an instance of the PlDataGranule for each item, ensuring that instance is not a duplicate already within the table.
6. If the stop time of the last returned granule is less than the stop time of the production request then the software predicts the data arrivals using the PlDataSource class.

4.5.3.8 Event Trace

See Figure 4.5-6.

4.5.4 Subscription Submission Scenario

4.5.4.1 Abstract

This scenario describes the submitting a subscription to a Data Server from the Subscription submission utility.

4.5.4.2 Interfaces With Other Subsystems and Segments

The Interoperability subsystem provides advertisements to the Planning subsystem. The subscription is submitted to the Data Server subsystem. The MSS provides user information required to set up a subscription.

4.5.4.3 Stimulus

The production scheduler initiates the Subscription Submission Interface in order to submit a subscription.

4.5.4.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubscriptionSubmitIF
- PlDataTypes
- PlDataType
- IoAdAdvertisingSrv_C

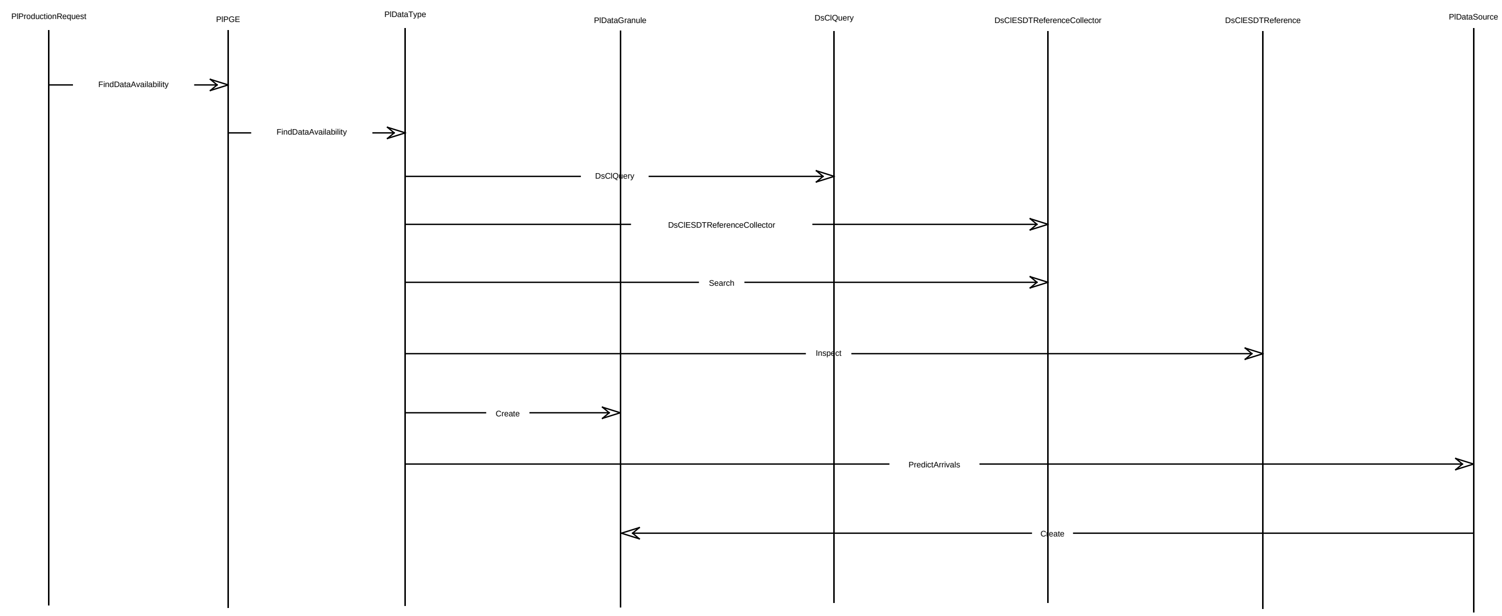


Figure 4.5-6. Data Availability Event Trace

- IoAdService Collection
- IoAdServiceAdvertisement

4.5.4.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.4.6 Ending State

No change in PDPS database server state. The subscription is submitted.

4.5.4.7 Scenario Description

1. The user starts the subscription submission utility, the standard user authentication process applies (see scenario 4.5.17). The user selects to set up a subscription and is offered a display of those Data Types which may be subscribed to.
2. The user selects the Data Type for which to set up a subscription.
3. The subscription submission utility selects the advertisement corresponding for the selected Data Type, using the Data Server UR and product name attributes of the Data Type class.
4. The user confirms the submission of the subscription.
5. The utility builds the Data Server subscription from the advertisement, submits the subscription and, on a successful status from the Data Server updates the Data Type object within the PDPS database.

4.5.4.8 Event Trace

See Figure 4.5-7.

4.5.5 Subscription Withdrawal Scenario

4.5.5.1 Abstract

This scenario describes withdrawing a subscription to a Data Server from the Subscription submission interface.

4.5.5.2 Interfaces With Other Subsystems and Segments

The Interoperability subsystem provides advertisements to the Planning subsystem. The subscription is submitted to the Data Server subsystem. The MSS provides user information required to set up a subscription.

4.5.5.3 Stimulus

The production scheduler initiates the Subscription Submission Interface in order to withdraw a subscription.

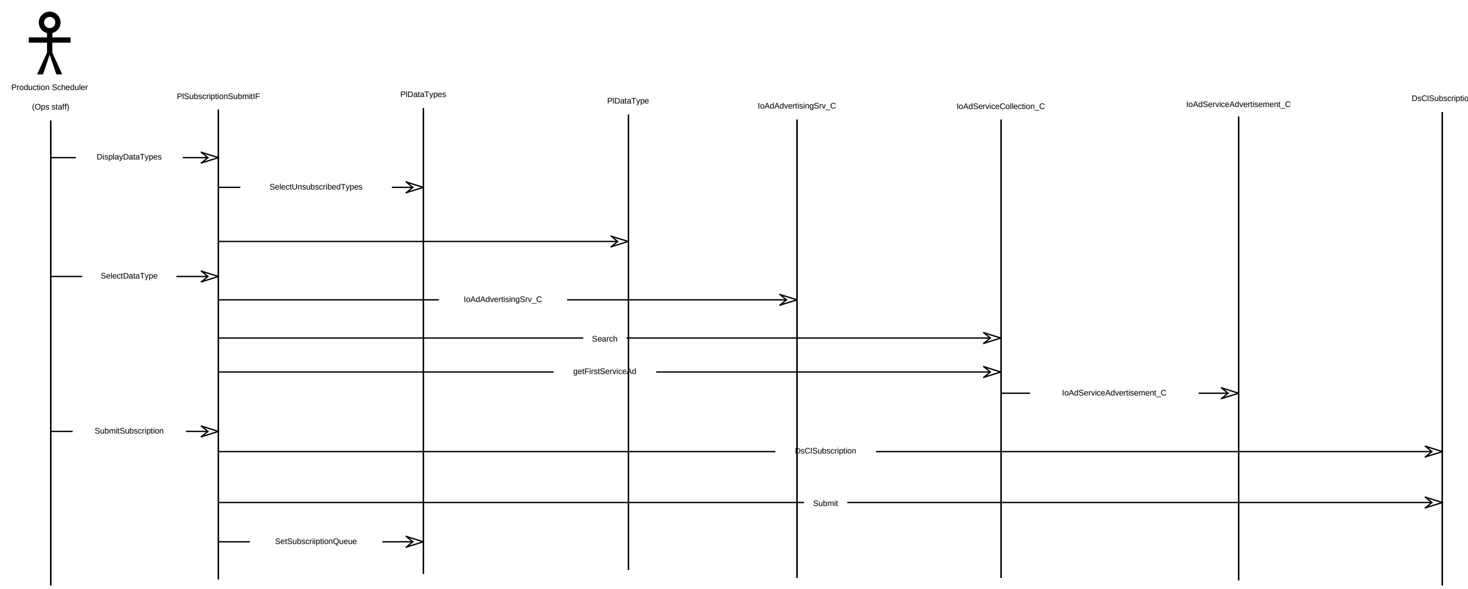


Figure 4.5-7. Subscription Submission Event Trace

4.5.5.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubscriptionSubmitIF
- PlDataTypes
- PlDataType
- IoAdAdvertisingSrv_C
- IoAdServiceCollection_C
- IoAdServiceAdvertisement

4.5.5.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.5.6 Ending State

No change in PDPS database server state. The subscription is withdrawn.

4.5.5.7 Scenario Description

1. The user starts the subscription submission utility, the standard user authentication process applies (see scenario 4.5.17). The user selects to withdraw a subscription and is offered a display of those Data Types which are subscribed to.
2. The user selects the Data Type for which to withdraw up a subscription.
3. The subscription submission utility selects the advertisement corresponding for the selected Data Type, using the Data Server UR and product name attributes of the Data Type class.
4. The user confirms the withdrawal of the subscription.
5. The utility builds the Data Server subscription from the advertisement, withdraws the subscription and, on a successful status from the Data Server updates the Data Type object within the PDPS database.

4.5.5.8 Event Trace

See Figure 4.5-8.

4.5.6 Ground Event Scenario

4.5.6.1 Abstract

This scenario describes the system response to a resource manager adding / modifying / deleting a ground event within the PDPS database. Ground events describe the allocation of a resource to a non-production task such as maintenance. In the majority of cases, these ground events will be defined well in advance of the production planning, and so a replanning of the production schedule is not part of this standard scenario. If the ground event is entered within a time period that is part of the activated schedule then, in order to have that event figure within the schedule, a replan will

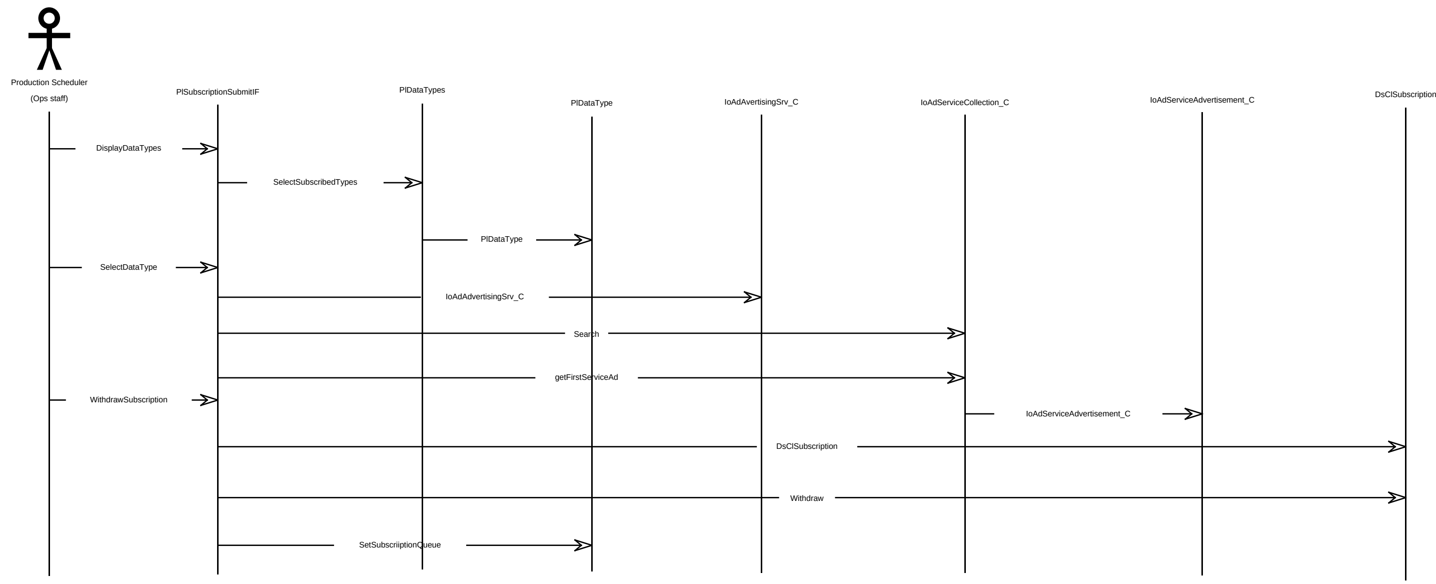


Figure 4.5-8. Subscription Withdrawal Event Trace

be required the resource manager (operations staff role) is required to communicate the need for a replan with the production scheduler (operations staff role) in order for that event to be scheduled.

Note: The description of a ground event allows a window of opportunity to be defined together with a duration. (So that the production can have some flexibility as to when to plan an event to achieve efficient resource usage). The actual time of the event will only be determined when the plan is generated, unless the duration is the same length of time as the window of opportunity.

4.5.6.2 Interfaces With Other Subsystems and Segments

None.

4.5.6.3 Stimulus

The resource manager initiates the Production Planning Workbench in order to enter a ground event.

4.4.6.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlResourceManagersUI
- PlGroundEvents
- PlGroundEvent

4.5.6.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.6.6 Ending State

No change in PDPS database server state.

4.5.6.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17).

Thread 1: Addition of a ground event

2. The user is presented with a function to add a new event, this initiates a forms type window (Event Description Window) in which that ground event can be described, the user populates the fields within the form, for event description, window of opportunity, duration, resource requirements etc.
3. The user confirms the description is complete and the Ground Event is created and recorded within the PDPS database. (Simple validation of the Ground Event will be performed to ensure the start date is before the end date etc.).
4. The timeline display of resources will be updated to show the allocation.

Thread 2: Modification of a ground event.

5. The user is presented with a function to modify an event described within the PDPS database, this initiates a window (Event Selection Window) which provides a list of the current defined events. The facility to reduce the list by limiting the search to a given time range is provided.
6. The user selects a ground event from the list, which then populates the fields in the Event Description Window.
7. The user modifies the description and confirms the modification of the Ground Event.
8. The timeline display of resources will be updated to show the allocation.

Thread 3: Deletion of a ground event.

9. The user is presented with a function to delete an event described within the PDPS database, this initiates a window (Event Selection Window) which provides a list of the current defined events. The facility to reduce the list by limiting the search to a given time range is provided.
10. The user selects a ground event from the list, which then populates the fields in the Event Description Window.
11. The user confirms the deletion of the Ground Event.
12. The timeline display of resources will be updated to show the allocation.

4.5.6.8 Event Trace

See Figure 4.5-9.

4.5.7 Plan Creation Scenario

4.5.7.1 Abstract

This scenario describes the creation of a plan within the Production Planning Workbench.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description please refer to the Production Planning Object Library CSC section (section 4.6.5).

4.5.7.2 Interfaces With Other Subsystems and Segments

None.

4.5.7.3 Stimulus

The production scheduler initiates the Production Planning Workbench in order to generate a new plan.

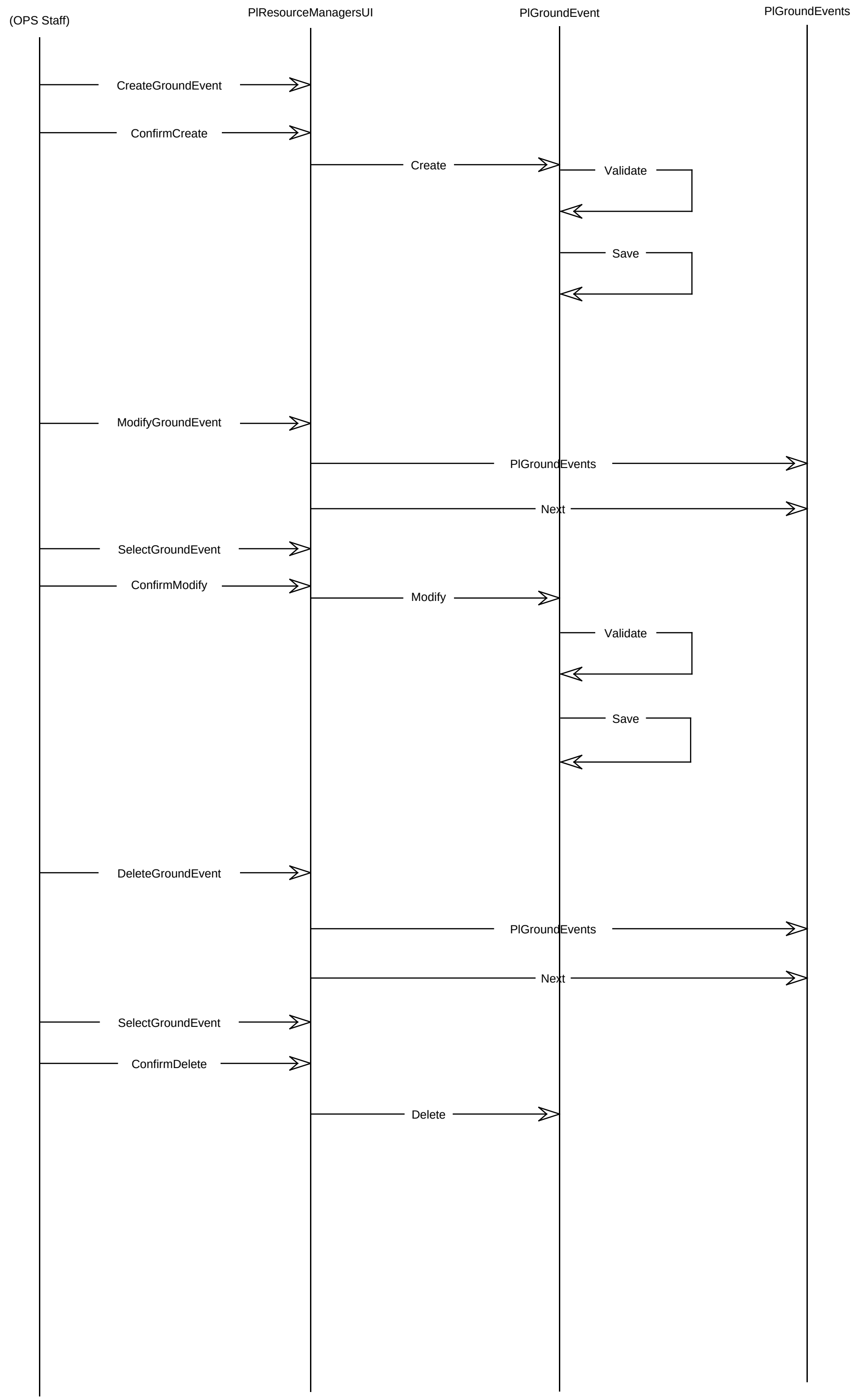


Figure 4.5-9. Ground Event Entry Event Trace

4.5.7.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundEvents
- PlGroundEvent
- PlGroundActivity
- PlDPRs
- PlDPR
- PlPGEActivity
- PlResourceManager

4.5.7.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.7.6 Ending State

No change to system state.

4.5.7.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17). The user specifies the time period for which to generate a plan within the GUI and initiates the creation of a plan object.
2. The plan object determines which ground events are defined within the duration of the planning time-period, by creating an instance of the PlGroundEvents collection object.
3. The plan object iterates through the ordered list of ground events, creating an activity to fulfil each event
4. The activity is allocated to the appropriate resources by the resource manager, thus setting out a timeline of when the resources are unavailable for production.
5. The user then specifies the Production Requests which are to be included in the plan.
6. The Data Processing Requests which are associated to the Processing Request, and within the time period of the plan are determined from the PDPS database using the PlDPRs collection class.
7. The plan object iterates through the ordered list of Data Processing Requests, creating an activity to signify the run of the PGE.
8. The activity is allocated to the appropriate resources by the resource manager.
9. The schedule for the PGE executions is redetermined by the plan object, defining the predicted start and predicted stop times of all the activities.

4.5.7.8 Event Trace

See Figure 19.

4.5.8 Deleting a Plan Scenario

4.5.8.1 Abstract

This scenario describes the deletion plan from the Production Planning Workbench. This function does not delete the associated DPRs or Ground Events from the PDPS database, since they may be associated to other plans, and are also maintained for a set period such that reports may be generated against their completion status.

4.5.8.2 Interfaces With Other Subsystems and Segments

None.

4.5.8.3 Stimulus

It's assumed that the Production Planning Workbench application is running. The user decides to delete the production schedule from the workbench.

4.5.8.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlPGEActivities
- PlPGEActivity

4.5.8.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The Production Workbench application is running and that a plan has been selected.

4.5.8.6 Ending State

No change in PDPS database server state. The plan is deleted.

4.5.8.7 Scenario Description

1. The operator initiates the command to delete the displayed plan.
2. The plan object creates an ordered list of the activities within the plan.
3. The plan object iterates through the activities and invokes the delete method from the activity [note that the delete method means that the activity is removed from the PDPS database, as opposed to the destructor operation for the class, which deallocates the memory associated to the object within the application, the destructor is called after the delete].
4. The entry describing the plan itself in the database is deleted.

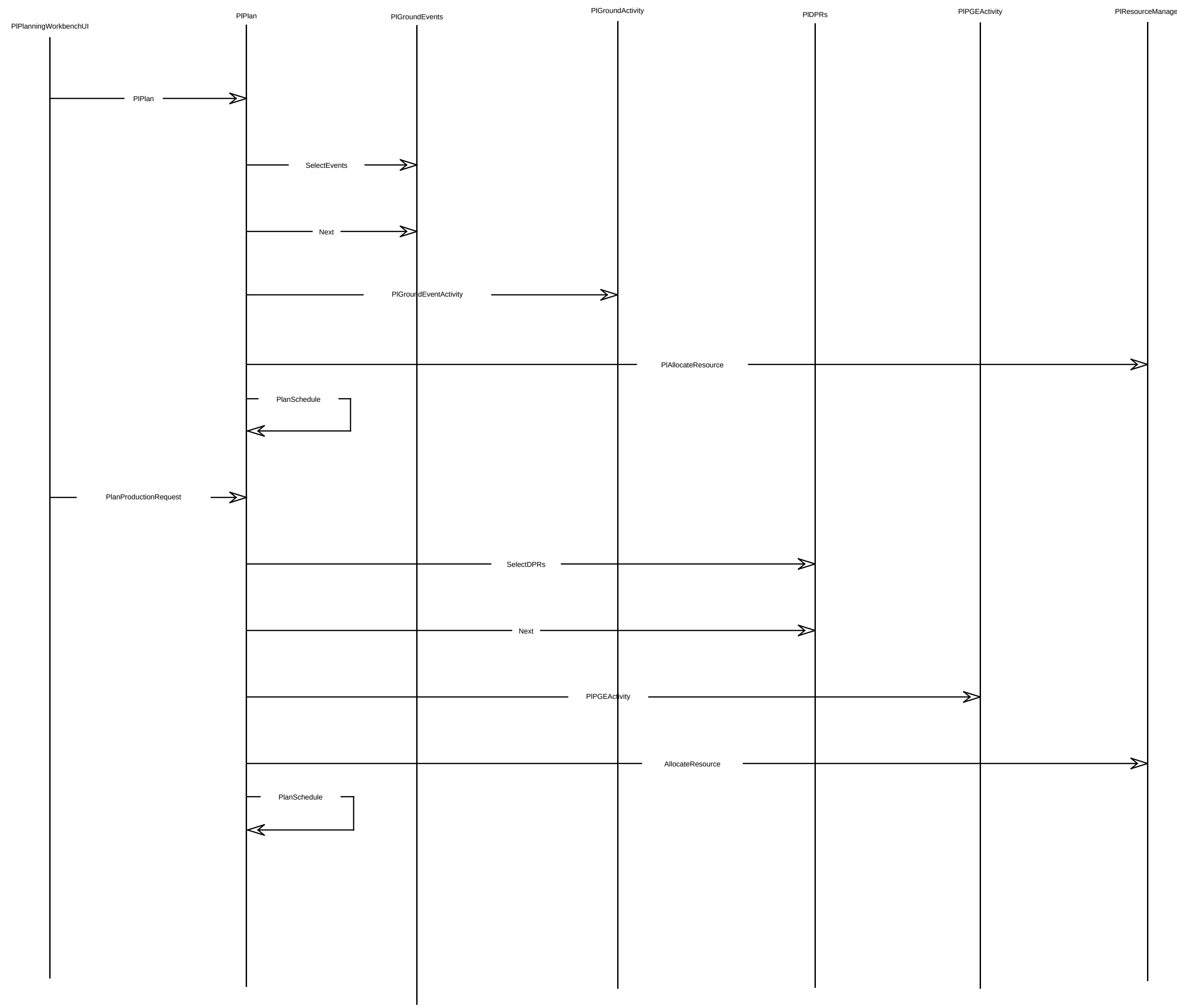


Figure 4.5-10. Plan Creation Scenario

4.5.8.8 Event Trace

None.

4.5.9 Publishing a Plan Scenario

4.5.9.1 Abstract

This scenario describes the system response to a production planner publishing a plan which is maintained within the PDPS database. After being selected, the plan would be formatted by the system into a published plan consisting of three different types of file, which are Metadata file, ASCII report file, and Binary report file. The published plan then would be insert into Documentation Data Server. The method to insert a published plan into Document Data Server is an operation encapsulated in the PlPublishedPlan Class.

4.5.9.2 Interfaces With Other Subsystems and Segments

Document Data Server provides storage for published plans. The interface to the Document Data Server is shown by the DsDoProductionPlan object, which is Document Data Server abstraction for the plans.

4.5.9.3 Stimulus

A production planner initiates PublishPlan Function from the Production Planning Workbench.

4.5.9.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlPublishedPlan
- PlPlanMetadataFile
- PlPlanASCIIReportFile
- PlPlanBinaryReportFile

4.5.9.5 Beginning System, Segment and Subsystem State(s)

Steady state, up and running.

4.5.9.6 Ending State

No change in the state of the system.

4.5.9.7 Scenario Description

1. The production planner starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17).
2. The production planner is presented with a function to publish a plan, and initiates the function.

3. Published Plan object is created.
4. The published plan is inserted into Documentation Data Server.
5. Planning System will send a notification by mail to a pre-prepared list of address when a new plan is published.

4.5.9.8 Event Trace

See Figure 4.5-11.

4.5.10 Building the Resource Configuration Scenario

4.5.10.1Abstract

This scenario describes the initialization of the Resource Configuration from MSS

4.5.10.2Interfaces With Other Subsystems and Segments

The MSS provides resource configuration information to the subsystems.

4.5.10.3Stimulus

The resource configuration is built, or updated as a manual initiated operation from the Planning Workbench.

4.5.10.4Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
  PlResourceConfiguration
- PlResource
- PlString
- PlComputer
- PlDiskPartition
- MsDAAC

4.5.10.5Beginning System, Segment and Subsystem State(s)

Steady state, up and running.

4.5.10.6Ending State

No change in the state of the system.

4.5.10.7Scenario Description

1. The production planner starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17).
2. The production planner is presented with a function to build the resource configuration, and initiates this function.

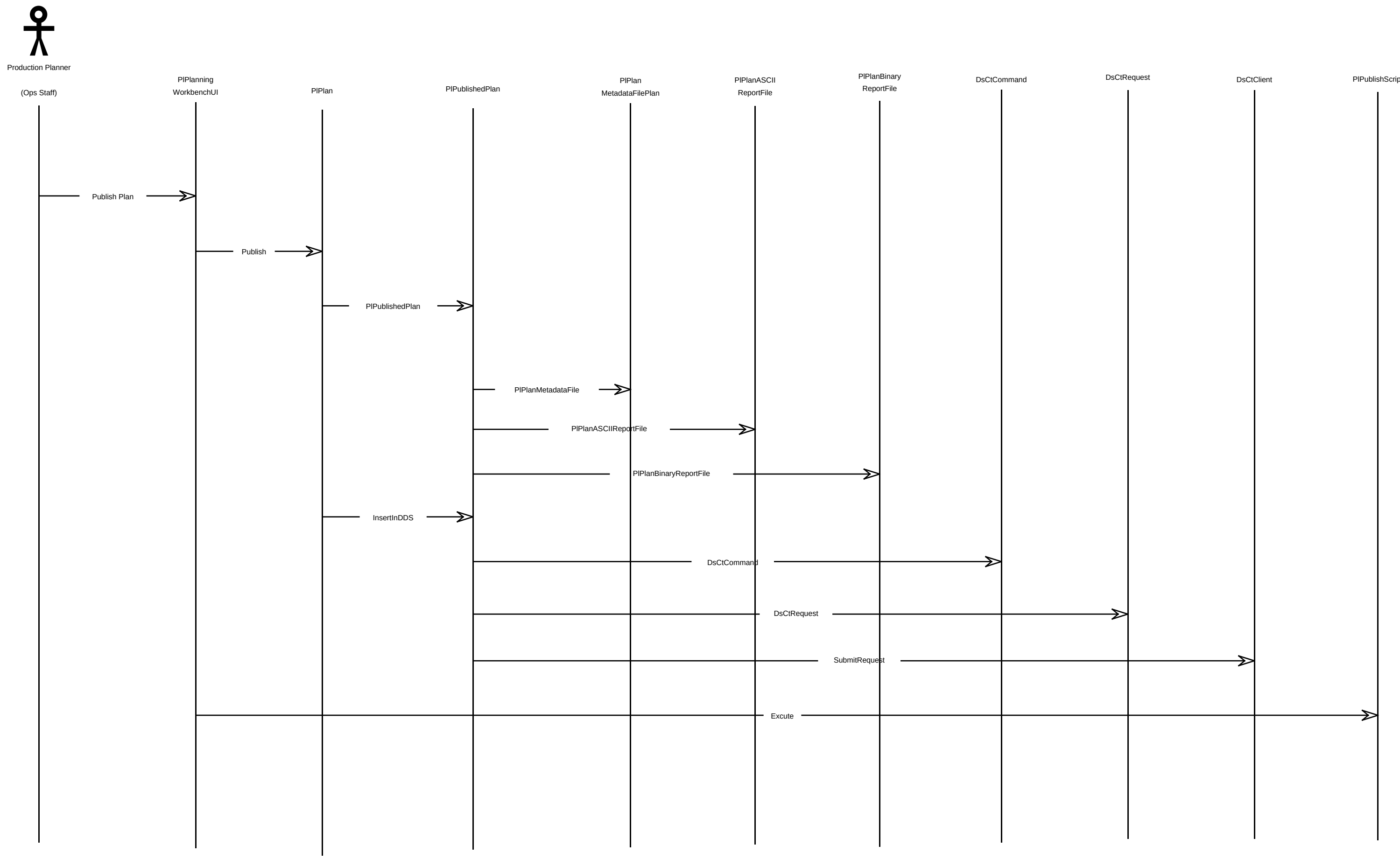


Figure 4.5-11. Plan Publication Event Trace

3. MsDAAC object is created, and a filter applied to specify the need for information on production resources only.
4. The iteration services of the MsDAAC class are used to extract the production resource and create the objects for these within the PDPS database.

4.5.10.8 Event Trace

See Figure 4.5-12.

4.5.11 Plan Activation Scenario

4.5.11.1 Abstract

This scenario describes the activation of a plan from the Production Planning Workbench.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description of the “planning/scheduling” aspects of this scenario please refer to the Production Planning Object Library CSC section (section 4.6.5).

4.5.11.2 Interfaces With Other Subsystems and Segments

None.

4.5.11.3 Stimulus

The production scheduler initiates the Production Planning Workbench in order to activate a plan.

4.5.11.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundActivies
- PlGroundEvent
- PlPGEActivities
- PlDPR
- PlActivities
- DPJobScheduler

4.5.11.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. No previous plan is active (there are no scheduled jobs in the data processing subsystem's job scheduler COTS).

4.5.11.6 Ending State

No PDPS database server state. A plan is activated (a portion of the active plan has been rolled into the data processing subsystem's job scheduler COTS).

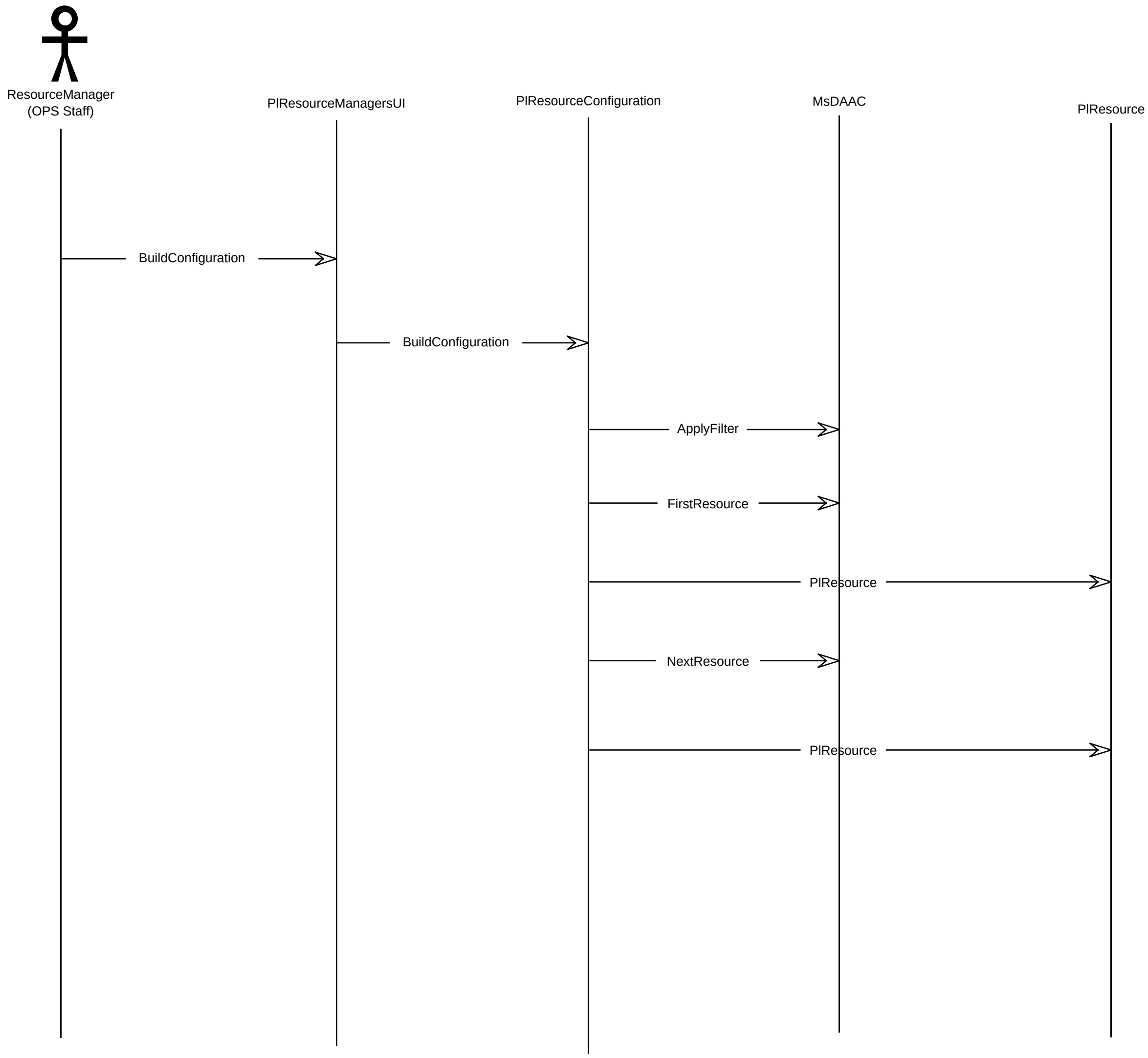


Figure 4.5-12. Resource Configuration Event Trace.

4.5.11.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17). The user selects a plan which is to be used to drive the production schedule.
2. On recreation from the PDPS database the plan is automatically updated to reflect changes that have occurred since the plan was generated.
3. The plan object determines whether any new ground events have been defined, or old ground events have been deleted since the plan was generated or last updated. The Activities describing the ground events within a plan are updated accordingly and the plan is updated.

4. The plan object determines whether any of the data processing requests within the plan have been completed and updates the plan accordingly. The user may modify the plan at this point by redefining priorities in order to achieve a revised schedule, this thread is not described here.
5. The user initiates activation of the plan. The plan object will create an ordered list of the activities which are within the scheduling window. This window defines the portion of the plan that needs to be rolled into the COTS. By default this window is 24 hours from the current time, however it is modifiable within from the workbench utility.
6. The plan object iterates through the activities defined in the order indicated by the plan, and calls the appropriate scheduling operation for that activity.
7. The Data Processing Request activities will initiate a call to create a PGE job within the data processing subsystem's job scheduling COTS.
8. The Ground Events will initiate a call to create a Ground Event Job within the data processing subsystem's job scheduling COTS.

Note: For details of how the jobs are created within the COTS and subsequently managed from the COTS please refer to scenarios within the Data Processing Design documentation.

4.5.11.8 Event Trace

See Figure 4.5-13.

4.5.12 Cancelling a Plan Scenario

4.5.12.1 Abstract

This scenario describes the cancelling of an active plan from the Production Planning Workbench. The scenario illustrates the interface to the Data Processing subsystem through the DPJobScheduler interface class.

4.5.12.2 Interfaces With Other Subsystems and Segments

The Planning subsystem interfaces with the Data Processing subsystem in order to cancel the scheduled Data Processing Requests.

4.5.12.3 Stimulus

It's assumed that the Production Planning Workbench application is running and displaying the active plan. The user decides to cancel the production schedule from the workbench.

4.5.12.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlActivePlan
- PlPGEActivities
- PlPGEActivity
- PlGroundEvent

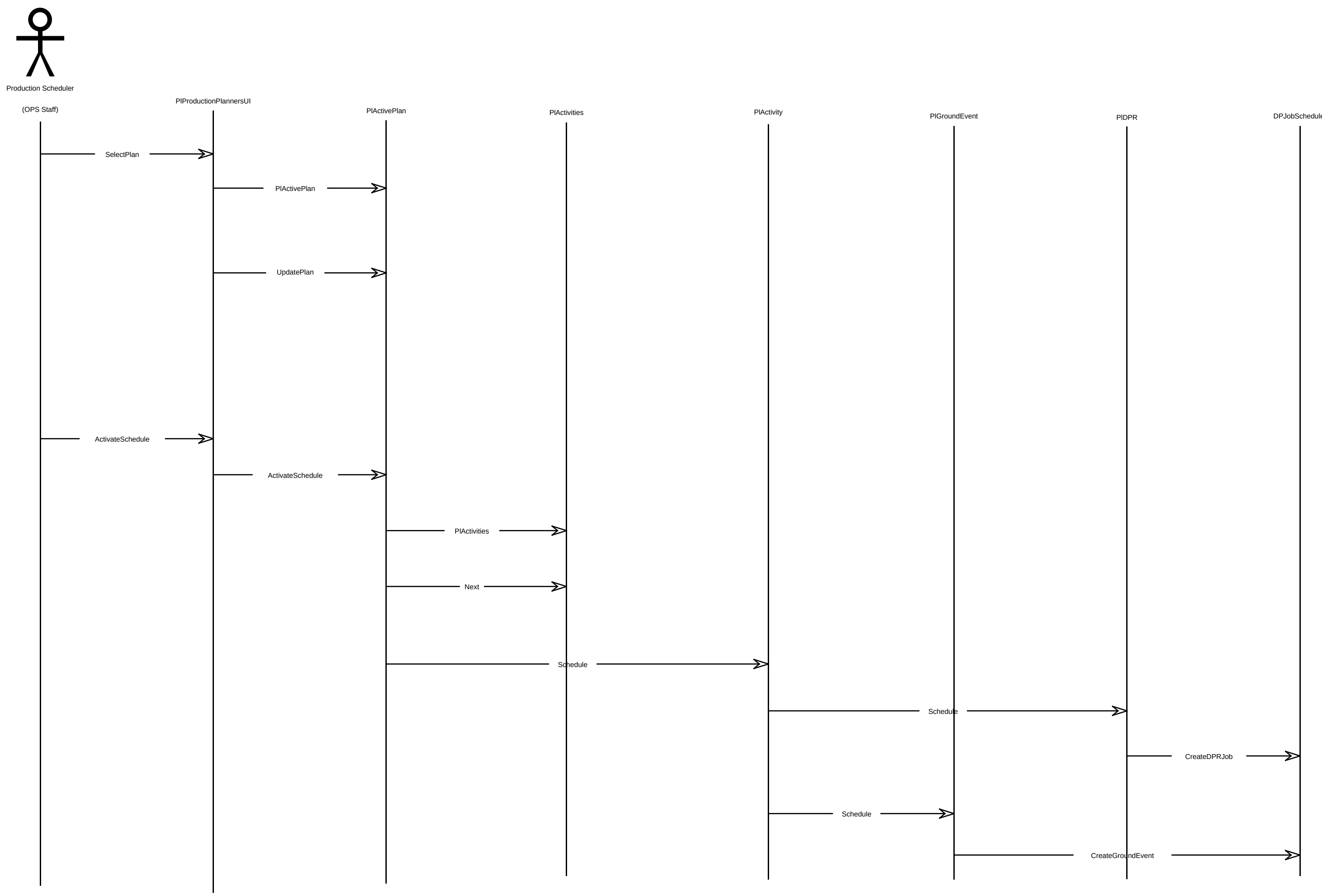


Figure 4.5-13. Plan Activation Event Trace

- PlDPR
- DPJobScheduler

4.5.12.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. A portion of a plan has been activated (there are scheduled jobs in the data processing subsystem's job scheduler COTS).

4.5.12.6 Ending State

No change in PDPS database server state. The scheduled activities are cancelled.

4.5.12.7 Scenario Description

4. The operator initiates the command to cancel the production schedule.
5. The plan object creates an ordered list of the activities which are part of the active schedule (determined from the status attribute).
3. The plan object iterates through the activities and invokes the cancel method from the activity.
6. The PlDPR and PlGroundEvent classes interface with the DPJobScheduler class to cancel the activity.
7. The plan is updated.

4.5.12.8 Event Trace

See Figure 4.5-14.

4.5.13 Statusing a Plan Scenario

4.5.13.1 Abstract

This scenario describes the statusing of a plan from the Production Planning Workbench. A component of the Job Scheduling COTS, AutoXpert now provides the dynamic display of the production schedule, therefore there are no requirements for dynamic updating to the plans. However there are occasions when the Plan from which the schedule originated would require updating, or statusing. Primarily the need for updating this will be before “downloading” a day's schedule into the COTS, taking into account the previous days production.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description of the “planning/scheduling” aspects of this scenario please refer to the Production Planning Object Library CSC section (section 4.6.5).

4.5.13.2 Interfaces With Other Subsystems and Segments

None.

4.5.13.3 Stimulus

The production scheduler initiates the Production Planning Workbench in order to activate a plan.

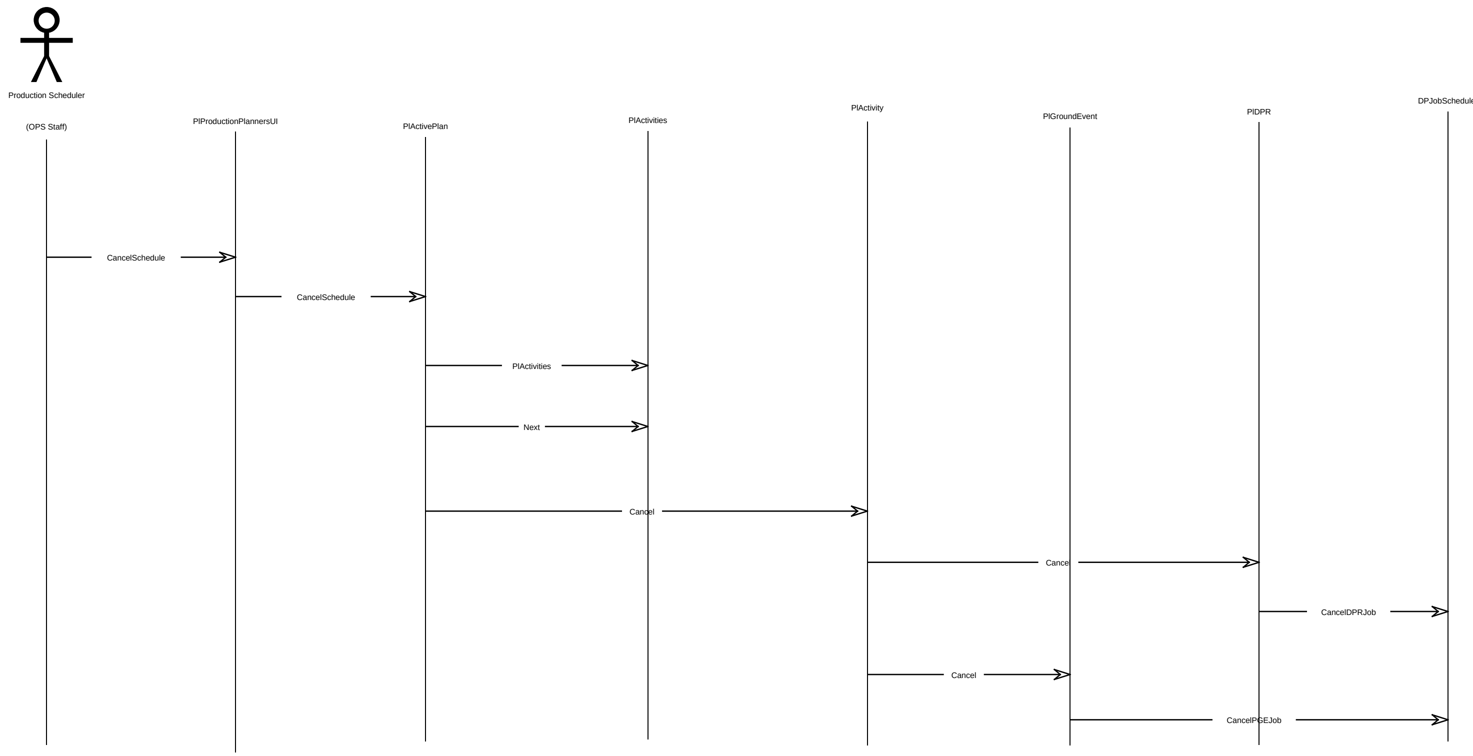


Figure 4.5-14. Plan Cancellation Event Trace

4.5.13.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundActivies
- PlGroundEvent
- PlPGEActivities
- PlDPR
- PlActivities
- DPJobScheduler

4.5.13.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. A portion of a plan has been activated (there are scheduled jobs in the data processing subsystem's job scheduler COTS).

4.5.13.6 Ending State

No PDPS database server state. The plan is updated.

4.5.13.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.5.17). The user selects the active plan which is to be updated.
2. On recreation from the PDPS database the plan is automatically updated to reflect changes that have occurred since the plan was generated.
3. The plan object determines whether any new ground events have been defined, or old ground events have been deleted since the plan was generated or last updated. The Activities describing the ground events within a plan are updated accordingly and the plan is updated.
4. The plan object determines whether any of the data processing requests within the plan have been completed and updates the plan accordingly.
5. The plan object creates an ordered list of the activities which are part of the active schedule (determined from the status attribute). For those activities which have not completed the plan object invokes the status method.
6. The PlDPR and object interfaces with the DPJobScheduler class to return the status of the activity.
7. The plan is updated.

4.5.13.8 Event Trace

See Figure 4.5-15.

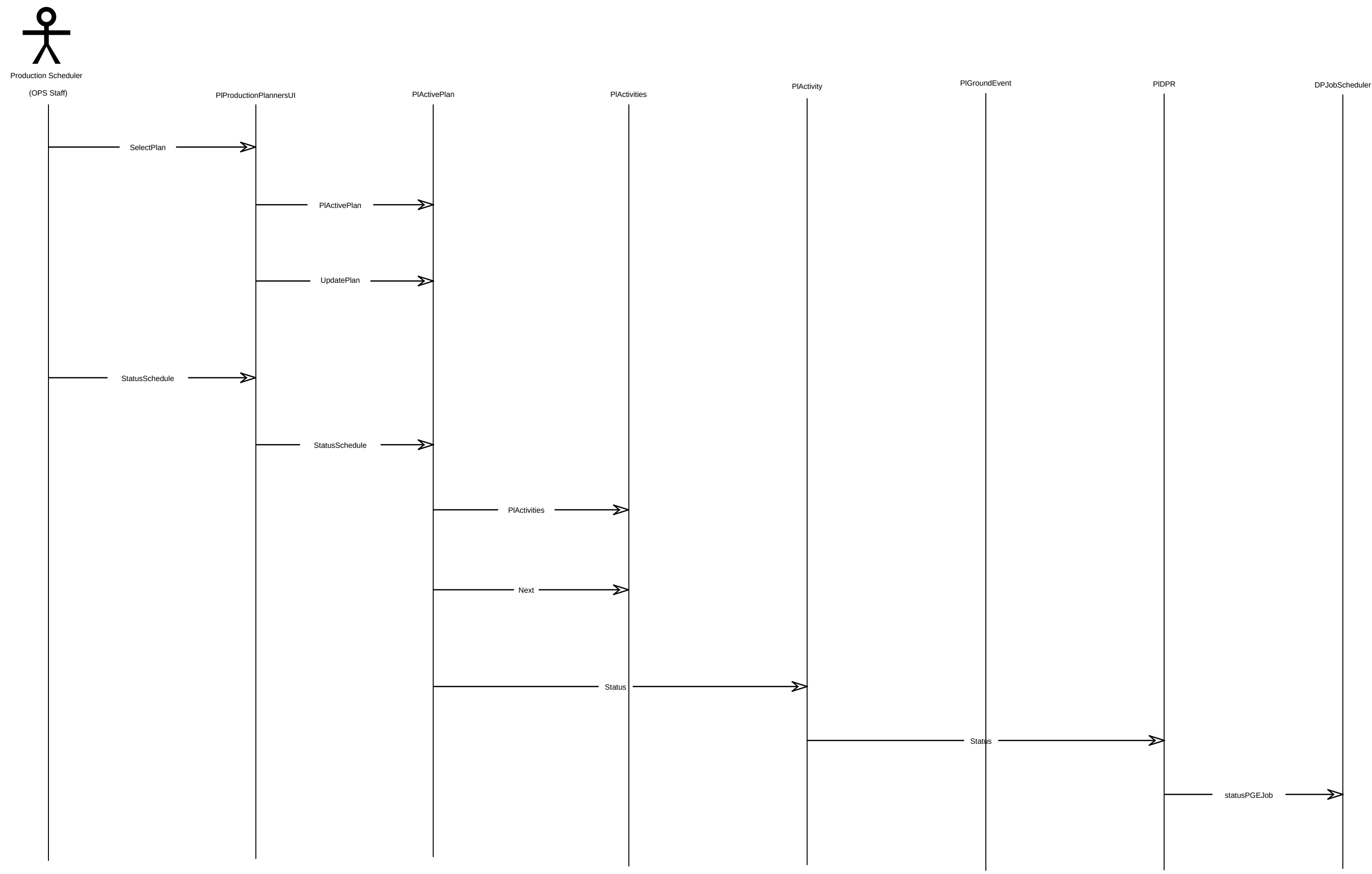


Figure 4.5-15. Plan Statusing Event Trace

4.5.14 Subscription Manager Startup Scenario

4.5.14.1 Abstract

This scenario describes the startup procedure for the Subscription Manager application. This scenario describes the activities within the Subscription Manager once the application is started, not the MSS activities which initiate the program. For a description of how ECS servers are started see Release A CSMS Management Subsystem Design Specification

4.5.14.2 Interfaces With Other Subsystems and Segments

MSS communicates lifecycle commands (startup, shutdown etc.) to ECS applications.

4.5.14.3 Stimulus

The ECS Operator starts the Subscription Manager from the Management User Interface.

4.5.14.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubscriptionManager
- MsManager
- PlSubscriptionManagerCallbacks
- CSS Notification
- PlDataTypes
- PlDataType

4.5.14.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is not running.

4.5.14.6 Ending State

No change in PDPS database server state. The subscription manager is started and waiting notification of data arrival from the Data Server.

4.5.14.7 Scenario Description

1. The Operator starts the subscription manager server.
2. The subscription manager declares itself as an ECS server application to the MSS subsystem and installs callbacks for shutdown and notification messages.
3. The subscription manager uses the PDPS database to determine those datatypes which have subscriptions declared with a Data Server, using the PlDataTypes class.
4. The subscription manager iterates through the subscribed data types and installs callbacks to be called whenever there is a subscription delivered from the Data Server.

Note: the guaranteed asynchronous nature of the subscription notification will mean that the application will be made aware of any subscriptions that have been delivered in the interval between it's last shutdown and the present startup. Scenario 4.5.15 describes the standard procedure performed each time a subscription notification arrives.

4.5.14.8 Event Trace

See Figure 4.5-16

4.5.15 Subscription Notification Scenario

4.5.15.1 Abstract

This scenario describes the procedure by which the subscription manager is informed of data arrival and performs the appropriate actions to determine whether or not to release one of the PGE jobs defined within the Data Processing subsystem

4.5.15.2 Interfaces With Other Subsystems and Segments

The Data Server subsystem initiates the subscription notification. The Planning subsystem releases PGE jobs within the Data Processing Subsystem

4.5.15.3 Stimulus

A notification of data arrival is sent from the Data Server subsystem.

4.5.15.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- CSSNotification
- PlDataTypes
- PlDataGranules
- PlDataGranule
- GlUR
- DsClESDTReference

4.5.15.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

4.5.15.6 Ending State

No change in PDPS database server, subscription manager state.

4.5.15.7 Scenario Description

1. The RegisterDataArrival callback is called, and the UR is created from the notification message.
2. The InspectDataArrival method is invoked to determine extract the metadata associated to the UR presented in the data arrival notification

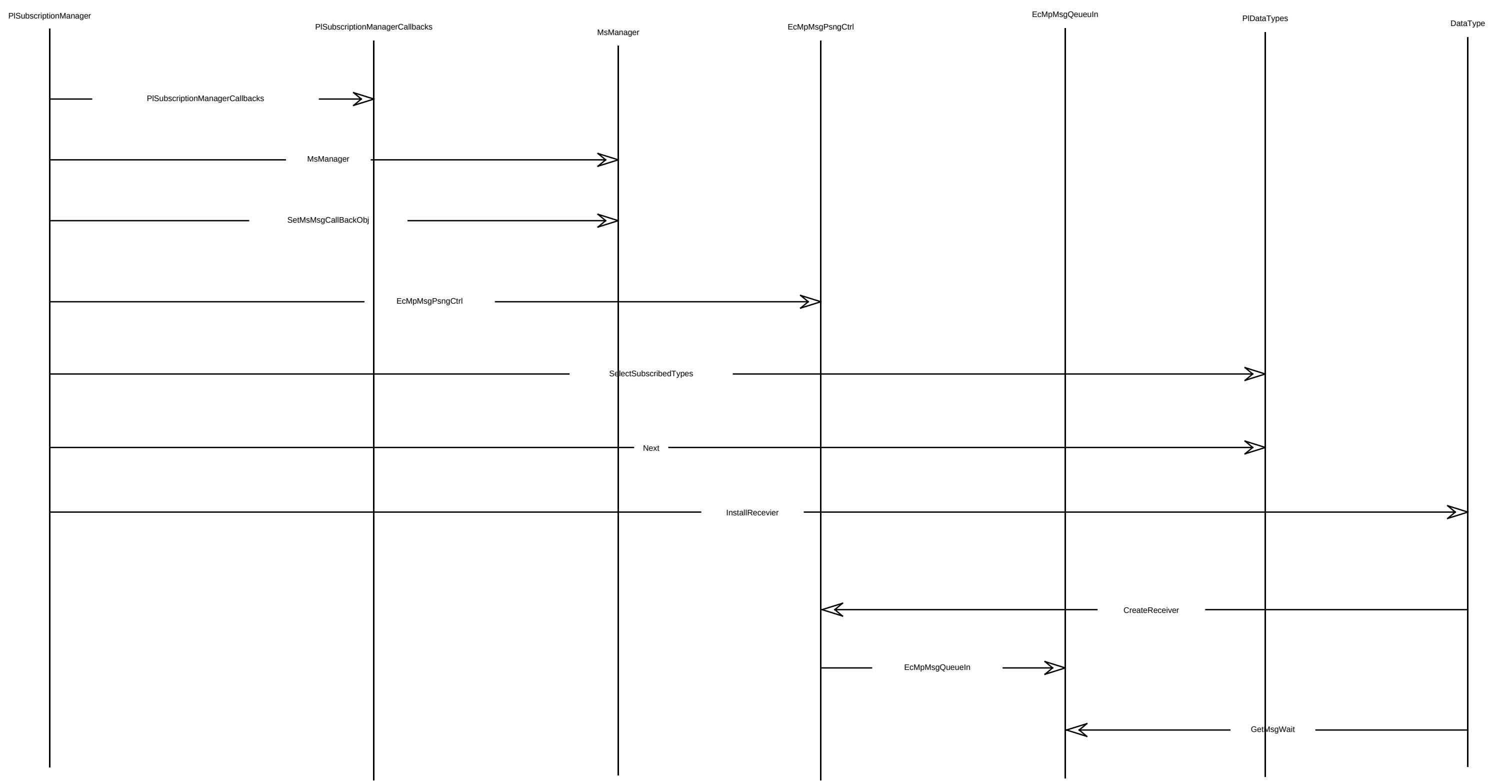


Figure 4.5-16. Subscription Manager Startup Event Trace.

3. The subscription manager determines the DPRs for which this data was required by selecting those predicted data instances which match the arrived data metadata.
4. The subscription manager iterates through the DPRs to determine whether all the data dependencies for the DPR have been fulfilled.
5. If the data dependencies have been fulfilled then the job associated with the DPR is released within the Data Processing subsystem

4.5.15.8 Event Trace

See Figure 4.5-17.

4.5.15.9 State Transition Diagrams

None.

4.5.16 Subscription Manager Shutdown Scenario

4.5.16.1 Abstract

This scenario describes the shutdown procedure for the Subscription Manager application. This scenario describes the activities within the Subscription Manager once shutdown notification is received, not the MSS activities which initiate the shutdown procedure. For a description of how shutdown of ECS servers is managed within MSS see the MSS Design Specification.

4.5.16.2 Interfaces With Other Subsystems and Segments

MSS communicates lifecycle commands (startup, shutdown etc.) to ECS applications.

4.5.16.3 Stimulus

The ECS Operator starts the Subscription Manager from the Management User Interface.

4.5.16.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubscriptionManager
- MsManager
- PlSubscriptionManagerCallbacks
- CSSNotification
- PlDataTypes

4.5.16.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

4.5.16.6 Ending State

No change in PDPS database server state. The subscription manager is shutdown.

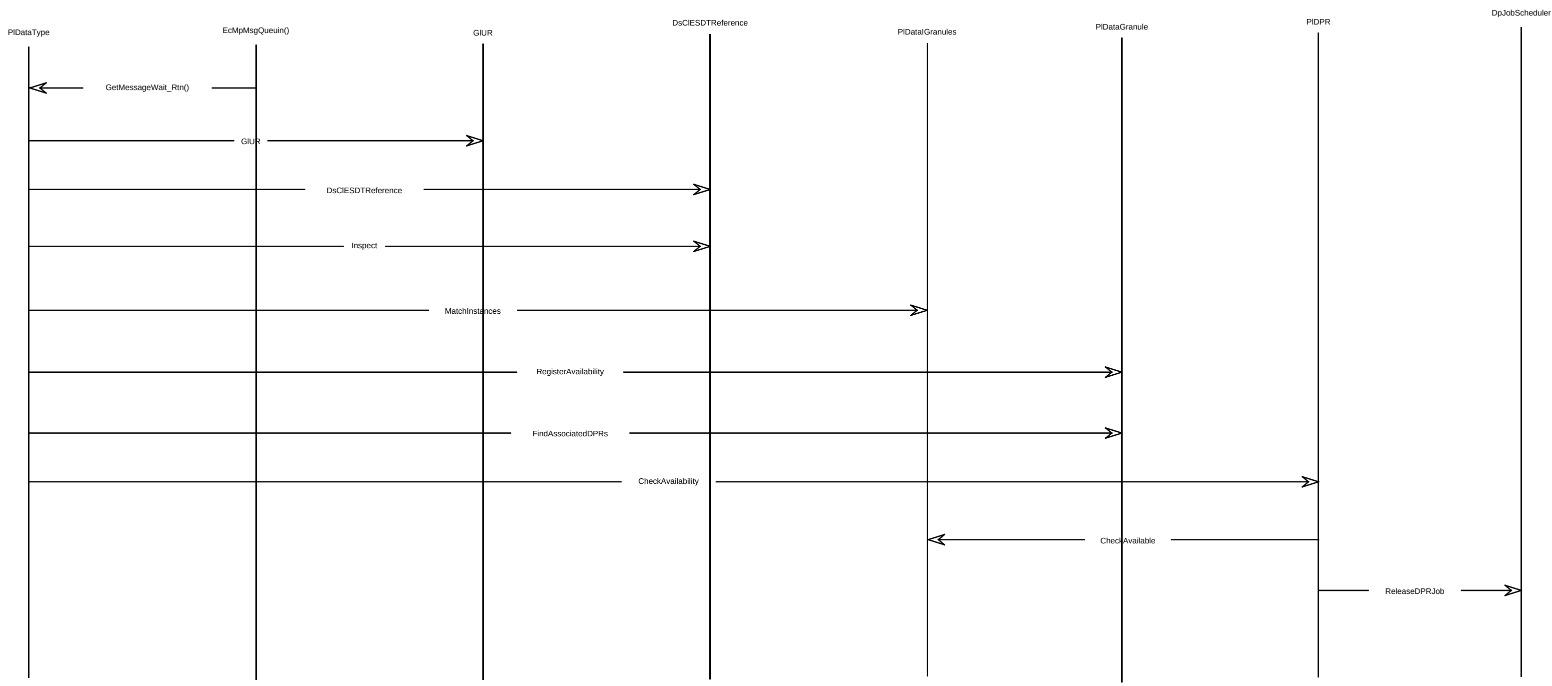


Figure 4.5-17. Subscription Notification Event Trace

4.5.16.7 Scenario Description

1. The PlSubscriptionManager callback is called by the MsManager class.
2. The PlSubscriptionManager shutdown procedure is called.
3. The subscription manager finishes actions with the outstanding notifications.
4. The subscription manager iterates through the subscribed data types and deinstalls callbacks.

Note: the guaranteed asynchronous nature of the subscription notification will mean that the application will be made aware of any subscriptions that are been delivered in the interval between it's shutdown and next startup. Scenario 4.5.15 describes the standard procedure performed each time a subscription notification arrives.

4.5.16.8 Event Trace

See Figure 4.5-18.

4.5.17 User Logon and Authentication

4.5.17.1 Abstract

This scenario describes user logon and authentication to access the Planning operational software. There are three classes of users, the production scheduler, resource manager, and the ECS user. The production scheduler and resource manager are operational staff primarily responsible for scheduling planning activities and managing resources respectively. The ECS user is primarily responsible for entering production requests.

The Planning software only permits users to access certain functional areas within the Planning software system. To this end, certain restrictions are place on different types of users. For instance, an ECS user cannot create a candidate plan whereas a production scheduler is allowed to create a candidate plan. These access rights are enforced through the Planning software and database. Users invoke the Planning software system, the account name is validated, and if authorized perform the day to day functions utilizing the operational system.

User accounts and associated privileges will be maintained in the Planning database by local DAAC administrative personnel. The Planning subsystem will provide a GUI interface to maintain the user accounts and privileges.

Note: Planning assumes that once a user has access to the ECS environment, i.e., has been authenticated by ECS, no password is necessary to access the Planning Software.

4.5.17.2 Interfaces with other Subsystems and Segments

None

4.5.17.3 Stimulus

The user logs into the Planning software and initiates the Planning software.

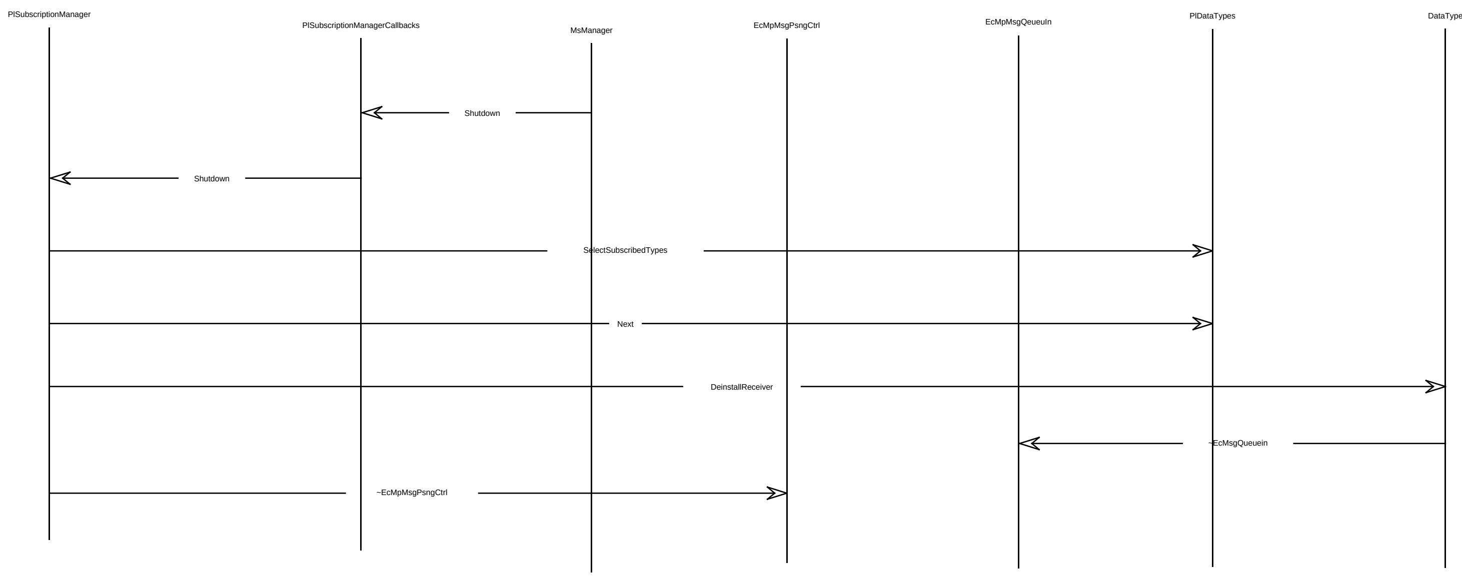


Figure 4.5-18. Subscription Manager Shutdown Event Trace

4.5.17.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlApplication,
- PlPlResourceManagersUI,
- PlPlanningWorkbenchUI,
- PlProductionRequestUI, and
- PlUser.

4.5.17.5Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

4.5.17.6Ending State

No change to the database server state.

4.5.17.7Scenario Description

1. The user logs into UNIX supplying an account name and password.
2. The user invokes the Planning software to gain access to its operational environment.
3. Upon initiation of the Planning software, the user's account name is captured.
4. Once the user account name is captured, the Planning software makes a connection to the database server to verify the account name.

Thread 1: Invalid User

5. If there is no match in the database for the corresponding account name, access will not be permitted to the Planning operational environment.
6. The Planning software notifies the user that he is not authorized to use Planning and all Planning services are denied.

Thread 2: Valid User

7. If the user is a valid user, the Planning software queries the database to obtain the appropriate privileges associated with the user.
8. The software assesses the privileges retrieved from the database and restricts the user to the access only the services within the system that are applicable to the user's class type.
9. Upon successful authentication the user proceeds to use the Planning software based on the user privileges retrieved from the database .

4.6 CSCI Structure

The CSCI is decomposed into a number of CSCs. The CSCs correspond either to an application, or a class category describing a logically related set of functionality. The table below briefly outlines the CSC breakout of the PLANG CSCI.

Table 4.5-1. Production Planning CSCs

CSC	Description	Type	
Subscription Editor	Interfaces to Data Server for submission of Subscriptions required so that Planning subsystem is notified of arriving data.	DEV	
Production Request Editor	Accepts Production Requests which describe an order for Data production, translates to Data Processing Requests which specify the PGEs which have to be run to service the Production Request.	DEV	
Subscription Manager	Interfaces to Data Server for notification of data availability, releases Data Processing Requests queued in Data Processing subsystem.	DEV	
Planning Workbench	Provides the ability to create, modify, and activate a plan for the scheduling of Data Processing Requests.	DEV	
Planning Object Library	A set of C++ class libraries that provides a framework for the Planning Workbench.	OTS	DELPHI (Hughes class libraries)
PDPS DBMS	DBMS Provides persistent storage for Planning data (e.g. PGE Profiles, Production Requests, Data Processing Requests, Plans etc.)	OTS	Sybase

4.6.1 Planning Subscription Editor CSC

4.6.1.1 Purpose and Description

This application provides the capabilities required to submit subscriptions to the Data Servers responsible for the storage of ingested data. Registration of a subscription at a Data Server is required for the Planning CSCI to receive notification when data arrive within the ECS. At this Release the submission of subscriptions will be managed as an operator initiated activity, although this may be automated at a later date.

4.6.1.2 Mapping to objects implemented by this component

As described in section 4.3.3.

4.6.1.3 Candidate products

None. This is a custom component required to interface with ECS specific services.

4.6.2 Production Request Editor CSC

4.6.2.1 Purpose and Description

This application allows the user to submit production requests that describe the data products to be produced. The application uses the PGE descriptions (profiles) entered during AI&T in order to work out the tasks - Data Processing Requests - that, in sum, meet the request. The application provides the capabilities to add, modify and delete Production Requests, as well as review and modify the resulting Data Processing Requests. The production request editor is identified as a distinct application and separate from the workbench in anticipation that defining production requests will be a discrete activity, unrelated to the “planning” of these events.

The Production request editor may also be used by authorized operations staff to schedule Data Processing Requests derived from a production request directly to the Data Processing subsystem.

4.6.2.2 Mapping to objects implemented by this component

As described in section 4.3.2.

4.6.2.3 Candidate products

None. Again is a custom component required to manage ECS specific entities such as the PGE.

4.6.3 Subscription Manager CSC

4.6.3.1 Purpose and Description

The Subscription Manager is used to manage the receipt of a subscription notification from the Data Server. Subscription notification is used to notify Planning on the arrival of input data required by a given PGE. The Subscription Notification contains Universal References (URs) which are pointers to the data objects stored in the Data Server. The Subscription Manager updates the PDPS database to indicate when data become available. When all input data for a Data Processing Request is available, the job defined for that Data Processing Request is released within the Data Processing subsystem.

4.6.3.2 Mapping to objects implemented by this component

As described in section 4.3.7.

4.6.3.3 Candidate products

This is a custom component required to interface with ECS specific services.

4.6.4 Production Planning Workbench CSC

4.6.4.1 Purpose and Description

The application is used to prepare a schedule for the production at a site, and forecast the start and completion times of the activities within the schedule. These functions provided by the workbench include the following high-level activities:

1) Candidate Plan Creation -- from the production requests prepared by the Production request Editor
2) Plan Activation -- activating a candidate plan
3) Updating the Active Plan -- feedback from the processing into the active plan.
4) Canceling/Modifying the Active Plan

As described previously, activating a plan entails rolling a portion of a selected plan into the AutoSys COTS. This “schedule” is then managed within the Data Processing subsystem. The forecast times generated within the planner are used to set up operator alerts that would make the operator aware of departures from the predicted schedule. The production planning workbench can periodically update it's predictions using feedback from the AutoSys.

4.6.4.2 Mapping to objects implemented by this component

As described in sections 4.3.4 to 4.3.6.

4.6.4.3 Candidate products

The Production Planning Workbench is build largely from COTS components. The planning workbench is built on top of a Planning Object Library that provides a framework for the application. This object library is provided by C++ class libraries which are described in the following section.

Job scheduling COTS are coming close to providing more capabilities that would meet some of the production planning capabilities, none of the packages at present meet the full requirements of the ECS. The AutoSys / AutoXpert packages are developing capabilities that at later releases capabilities may be suitable. The ECS is committed to influencing vendor direction to leverage capabilities to cover additional of the ECS requirements.

4.6.5 Planning Object Library

4.6.5.1 Purpose and Description

The Planning Object Library provides a reuse framework for build the Planning Workbench application. As described within the Planning Workbench CSC section 4.6.4, there is no complete “off the shelf” solution that covers all the Production Workbench requirements, and therefore reuse of robust class libraries is the best approach for developing this application. A number of planning and scheduling frameworks have been evaluated for scheduling in the Planning Object Library; these frameworks are described in the Scheduling Engine Evaluation Trade. In general, these libraries provide similar capabilities. The Hughes Delphi Scheduling Class Software has been selected from those considered since it consists of generic, non-application specific libraries and there is a high degree of expertise within the ECS with using these Libraries, both within the Planning subsystem through prototypes and within FOS where these libraries are used in the mission planning.

4.6.5.2 Mapping to Objects Implemented by this component

The CSCI object model presented in section 4.3.4 showed an abstract view of the Production Planning Workbench. This was mainly due to the fact that the complexity of the planning and scheduling aspects of the CSCI would overwhelm the rest of the model and scenario descriptions. It also the case that the capabilities could be implemented a number of different ways given different frameworks or COTS approach taken. The following sections first give an overall view of the Delphi scheduling class libraries and then show in detail how the Planning Workbench application is designed to reuse the Delphi libraries.

4.6.5.3 Hughes Delphi Scheduling Class Libraries

The Hughes Delphi Scheduling Software is designed to assist in developing efficient and effective scheduling and planning. It is composed of a set of user-oriented, integrated, modular tools.

The toolset provides building blocks to allow developers to:

- Model system resources that reflect all relevant operating states and constraints.
- Automatically create coarse or detailed schedules for all system resources based on requests, using a variety of tailorable scheduling algorithms.

- Provide high functionality interfaces for planners to review and edit service requests, corresponding resource information, and generated schedules.
- Provide support for interactive development of contingency or impact studies (*What-Ifs*).

Delphi is based upon a system of distributed, modular components. Each component represents a distinct planning function, and each component can be plugged in, disconnected, or replaced as changing concepts, system needs or software upgrades require. In addition, each component has been engineered utilizing object-oriented methodologies. This provides many significant benefits, including:

- Functionality - The software can be easily tailored to implement application specifics.
- Extensibility - Once delivered the system can be easily enhanced without the need for complete replacement or extensive and costly block changes.
- Maintainability - All functions (data and process behavior) are well encapsulated in the software architecture. Therefore, should a problem occur it can be easily identified and isolated. Metrics for maintenance indicate exceptional savings for customers in O&M costs for this software.

These features combine to support a high degree of system flexibility and expandability, and in the long term contribute to a significantly lower system life-cycle cost.

All of the tools in Delphi are built on a foundation called the Hughes Class Library (HCL). This product provides a framework for all the objects in the system and provides generic functions for services such as time, collections (list, sets, arrays, etc.), stream input and output, inter-process communication, and windows displays.

4.6.5.3.1 Delphi Resource Model

At the heart of Delphi is the Resource Model. In order to generate a timeline schedule for activities the Planning Workbench system must have detailed knowledge of both the activities that are to be performed and the resources that are required to be utilized or expended in order to complete each activity. Delphi's Resource Model supplies the structure to define these entities. In addition, the Resource Model acts as the owner of all data used by Delphi and therefore provides all data management services to the toolkit and the user.

The Resource Model consists of resources, resource states, and all relevant resource constraints.

Resources. All resources that are necessary for planning and scheduling are implemented as objects within the Resource Model. Application specific resource classes/objects can be derived from the resource classes provided in Delphi. These more specific classes/objects contain the attributes and behaviors that are unique to them.

Resource State. Besides modeling real world objects, another function of the Resource Model is to keep track of the state of all resources over time. Real-time and periodic (batch) updates of the state of all components of the system are sent to and stored by the resource model. For example, the state of a resource may indicate whether it is available or unavailable, and the nature of its current tasking.

Activities. In addition to resources and resource states, the resource model contains activities. These are schedulable entities that represent system tasking. During scheduling, resources are assigned to an activity. The resource's state is then updated to include each new activity. In turn, these updates can be directed as real-time modification to the timeline.

Constraints. Also in the Resource Model are the constraints present in the planning system. The scheduling algorithm will consider constraints between resources when attempting to properly allocate resources in a given plan.

The Resource Model defines, in both data structure and functional behavior, the resources being utilized. Any constraints concerning reasonable, proper, of safe behavior are defined to the system. The Resource Model also retains control over the definition of activities. These activities define a sequence of operations that are required to perform a high level goal. Both elements to scheduling, requests and resources, are managed within the flexible architecture of the Resource Model. Using these pools of information, the scheduling algorithms optimize the application of activities to resources to develop constraint-free plans for operational use.

Resource Model Hierarchies

There are two primary hierarchies within the Resource Model: whole/part activities and resource activities. The whole/part hierarchy provides an aggregation of many associated system objects; for instance, a "whole," such as the plan, contains "parts," such as resources and activities. In this manner. the Resource Model relates its individual inheritance hierarchies to create a global, consolidated, constraint-free plan.

A second hierarchy involves the individual resources and activities with which the planning and scheduling system must deal. In general, a resource hierarchy provides a detailed model of all relevant system resources. The activities are defined as resource-independent descriptions of operations that ultimately will be assigned to resources at specific times.

Each resource is responsible for maintaining a record of its state through time, for exporting algorithms of general interest, and for modeling and applying constraints. The resource state models what the resource is doing through time and may be different for each kind of resource. Resources know what plans are available and can have a different state on different plans.

Use of Resource State

Clients of the resource model can interact with resource state in several ways. A client can interact with resource state directly by asking a resource for all state on a given plan for a given time interval. The client would then have a detailed knowledge of the resource and can have as much knowledge of the resource and its inner workings as appropriate.

A client can also ask a resource for time intervals in which particular constraints are satisfied. These intervals may be displayed on another process, for example a timeline, in order to communicate to the user the window of opportunity for the resource. To obtain these intervals, the resource would iterate through all of its states and check with compatibility against existing state.

Resources can have a heterogeneous state. For examples, a resource could be tasked for a time interval and also be unavailable for another time interval. Both of these cases are supported by the resource state mechanisms.

Scheduling Activities to Resources

Resources that can be tasked can generally be asked to allocate (check constraints), unallocate (remove tasking), force allocate (do not check constraints), check allocate (check constraints but do not change state), and when allocate (tell me when constraints, if ever, are satisfied for a given set of tasking). This functionality is generally used by schedulers. For example, a sequential scheduler could order activities and then try activity/resource combinations until the activity was scheduled or until no combinations were left. The sequential scheduler will then move on to the next activity. Activities can be ordered by user-set priority, laxity, availability, or any other mechanism. The important point is that the scheduler only organizes the activities and orders the requests for resources. The resources themselves know what constraints to check and how to generate resource state. The scheduler may have to have resource-specific knowledge during the ordering process, or may use heuristics to optimize utilization of a resource.

4.4.5.3.2 Delphi Architecture

The following section describes the software architecture used by Delphi. Delphi is composed of a series of modular *libraries* of software. Each library defines a series of groupings of both data and functions that are referred to as classes. The classes serve to provide the mechanism for data structures and manipulation functions to be encapsulated or localized, thereby providing discrete functionality that can easily be developed and debugged. These objects provide the atomic functionality to all Delphi tools.

4.6.5.3.3 Hughes Class Library

The Hughes Class Libraries form the core from which all scheduling products are developed, thus providing real cost savings in development, test time, and maintenance costs. The Hughes Class Libraries are implemented in C++ according to the current standard, as set forth in the AT&T Version 2.0 C++ Programming Language Standard. The bottom layer of software usage is composed of X-Windows code, per the X11 standard. HCL utilizes X11 and allows the user application code to make direct calls to X11, if necessary. Within HCL is a display class of utilities that support the Motif display standards. These libraries have been in existence since 1990, and have been thoroughly tested and fielded in several operational systems.

The Hughes Class Library (HCL) is a library of C++ class declarations. These declarations are general purpose programming utilities, and include:

- display classes (XView, XGL, and Motif)
- collection classes
- Inter-process communication classes
- other miscellaneous utilities (e.g. string, rectangle, command line, etc. classes)

HCL contains several libraries: misc, dispx, mdisp, and ipc. These are summarized below:

misc Library

The 'misc' library supports the collections in linked list array, or set format. It provides iterators over each type of collection for ease of movement throughout the collection. Collections can safely have multiple concurrent consumers. HCL allows classes derived from a common base class (HObject) to be stored in the same collection, thus allowing heterogeneous collections.

The choice of collection type (array, list, set) does not affect the application code. These collection types all are derived from a common class that established the protocol for the derived classes.

This library also contains classes that provide various date/time functions, command line information, timer functions, string functions, and vector and matrix functions.

dispx Library

The 'disp' library includes classes for drawing using XGL utilities. It has classes that encapsulate XGL contexts, rasters, fonts, and color usage. It also has classes to maintain display regions and collections of sub-regions.

mdisp Library

The 'mdisp' library supports displays with the look and feel specified in the Motif Style Guide. The library uses the Motif toolkit from the Open Software Foundation (OSF). The Motif toolkit is based on the X-Toolkit Intrinsics (Xt), which is the standard mechanism on which many of the toolkits written for the X-Windows System are based. The user will notice that the library encapsulates capabilities at all three levels (i.e. Motif, Xt, and Xlib).

The library provides classes that allow the user to create and manipulate X-Windows, color maps, events (i.e. keyboard, mouse buttons, window enter/exit), and user interface objects called widgets (i.e. menu bars, pulldown menus, buttons, scrollbars). There are classes that provide displayable regions, subregions, rectangles, scalable fonts, strings, and colors. Mdisp also provides a class implementation of the graphics context (GC).

The DAppl class provides a template for a display application. It provides the behaviors to create, run, and destroy the application. The class does not provide a base frame.

ipc Library

The 'ipc' library contains classes that encapsulate interprocess communications, providing a simple interface for the programmer. It is an implementation of Berkeley sockets and XDR streams.

Messages derived from a common message class (HMessage) can be passed between processes without the ipc code knowing anything about the contents of the message. This isolates the code that does know about the message content, making for a much easier, faster development and easier maintenance.

4.6.5.3.4 Delphi Reuse Libraries

The framework for all scheduling applications is HCL. In addition to this core product are reusable libraries of classes for virtually all aspects of a scheduling system. For purposes of discussion, these libraries will be grouped into four categories. These categories are: Scheduling, Resource, Timeline, and Planning Class Libraries. All libraries build upon the base of the HCL and are developed in the C++ programming language.

Scheduling Class Library

The Scheduling Class Library provides a framework for the incorporation of scheduling algorithms.

Reusable Classes:

SResource (class modeling resources with scheduling) - Each SResource specifies protocol for allocation of an activity to a resource for a time interval.

SRsPool (scheduling resource pool) - An instance of this class provides a storage mechanism for quick retrieval of scheduling resources (indexing by associated resource ID). Derived classes can provide special queries that are typically process non-specific. For example, fill a given collection with all scheduling resources of a given type that have no state for a given time interval.

SSimpRs (class modeling resources with mutual exclusion) - Each SSimpRs is associated with an RResource and implements allocation members that assume a mutual exclusion constraint.

SUpdCatAbs (resource model change notifier) - SUpdCatAbs is a base class that sets up protocol for registration of changes of resources, their states, activity changes, and plan changes. It also has protocol for flushing these change notices. Derived instances of this class might keep lists of clients interested in changes, and notify these clients when changes take place.

SaActPriSrtr (activity priority sorter) - This class customizes the sorter for activity priority.SaAllImpct (Impact Scheduler) - Instances of SaAllImpct know how to accept lists of activities and edit the resource model accordingly, using impact scheduling. The plan that is edited is the one already set by using the SaComponent member function plan().

SaAllNImpct (Non-Impact Scheduler) - Instances of SaAllNImpct know how to accept lists of activities and edit the resource model accordingly, using non-impact scheduling. The plan that is edited is the one already set by using the SaComponent member function plan().

SaAllocator (activity allocator) - Abstract class used to define generic protocol for allocating schedulers.

SaArySorter (array sorter) - This class uses the system sort algorithm to perform sorting.

SaComponent (scheduling component) - Abstract class used to define generic protocol for scheduling algorithm components (sorters, filters, allocators, refiners and post processors).

SaFastFit (fast fit allocator) - This schedule establishes protocol for fast fit scheduling. The algorithm used is a first come, first serve algorithm.

SaFilter (allocation filter) - Abstract class used to define generic protocol for filtering an area of a plan.SaInflator (allocation inflation post processing) - This scheduling component class tries to inflate allocations.

SaPostProc (allocation post processor) - Abstract class used to define generic protocol for post processing schedulers.

SaRefiner (allocation refiner) - Abstract class used to define generic protocol for refining an area of a plan.

SaSorter (activity sorter) - Abstract class used to define generic protocol for sorting a collection of activities.

SaSurgStrike (surgical strike allocator) - This schedule establishes protocol for surgical strike fit scheduling. The consumer can provide an actual allocation algorithm.

Resource Class Library

The Resource Class Library provides the structure for defining application domain resources and it builds upon both HCL and the Scheduling Class Library. The Resource Class Library provides templates to define resource models capable of managing discrete resource states and mechanisms for assigning resource availabilities to resources. In addition, inter-request correlations, such as pre-requisite, co-requisite, and post-requisite constraints, preferences, consumable resource modeling, and specific configuration requirements, are addressed. Each of these resource definitions will be used by the schedule deconfliction processes to provide a constraint free plan. When an activity is defined, multiple resources can be identified as being applicable to that activity.

Reusable Classes:

RActAll (information about an activity allocation) - This class adds an activity to the base allocation. An example use is as a base class for modeling allocation of an activity to a bunch of resources. In that case, the derived class would have explicit resource support.

RActIdFactAbs (Unique activity id factory) - Instances of RActIdFactAbs are activity id generators. You can ask them for the next allowable activity id that can be assigned. This class has the notion of a global RActIdFactAbs object and provides a static nextId member function so that you do not have to have a specific instance everywhere you want to use it.

RActIdFactMem (Unique activity id factory) - Instances of RActIdFactMem are activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool.

RActPool (activity pool) - An instance of this class provides a storage mechanism for quick retrieval of activities (indexing by activity ID). Derived classes can provide special queries that are typically process non-specific. For example, fill this collection with all activities of a given type that have been allocated on a given plan.

RActState (tasking on a resource) - RActState is a resource state class that is generated by an activity.

RActivity (scheduling activity) - This is a base class for scheduling activities. It establishes protocol for all derived activities.

RAll (information about an allocation) - Each allocation has a time interval an optional plan name and a lock. If the plan name is empty, the allocation can pertain to all plans.RComplexAct (collection of activities) - Each instance of this class holds a collection of activities.

RDegState (tasking on a resource) - RDegState is a resource state class that represents a degraded resource. Presence of a direct instance of RDegState means that a resource is broken. Derived classes can represent how a resource is degraded.

RPlan (plan) - This is a base class that models the allocations of plans to resources.

RPlanPool (plan pool) - An instance of this class provides a storage mechanism for retrieval of plans (indexing by plan name). Derived classes can provide special queries that are typically process non-specific. For example, fill a given collection with all of the activities in the activity pool that have different allocations on two plans.

RResource (class modeling an entity with state through time) - RResource is a class that models an entity that has state through time. State is kept in RRsStateLists. Each RResource may have more than one RRsStateList. These state lists contain information which is mapped to a particular name (a “plan”). State lists usually contain things derived from RRsState. In other words, each resource object may maintain state information for multiple plans simultaneously. There is an additional, plan-independent state list that each RResource maintains. This state list may be used to hold plan independent information such as state that applies across all plans. RResource objects maintain a notion of a “current” state list.

RRsPool (resource pool) - An instance of this class provides a storage mechanism for quick retrieval of resources (indexing by resource ID). Derived classes can provide special queries that are typically process non-specific. For example, fill a given collection with all resources of a given type that have no state for a given time interval.

RRsState (base class for resource state) - Abstract class used to establish protocol for any kind of resource state.

RRsStateList (resource state list) - An HEpochIntervalList with a name. An important difference between this class and an HEpochIntervalList is that this class empties() its contents upon destruction (unlike a vanilla HObjCollection, which clears() it's contents).

RSchOpp (a scheduling opportunity) - Each scheduling opportunity has a time interval and a resource id, representing an possible allocation of an activity to that resource and interval.

RSchRqst (resource scheduling request) - This class is a list of scheduling opportunities, which are specified as instances of RSchOpp, with an activity id specifying the activity to be allocated using the opportunities and a parameter indicating how many of the opportunities must be satisfied for the entire scheduling request to be satisfied. Only derivations of RSchOpp should be added to this class, and the add behavior of HObjList has been overridden to enforce this.

RSimpleAct (simple scheduling activity) - Each instance of this class models something happening on a resource in the system. Simple activities add allocations to the concept of Activity.

Timeline Class Library

The Timeline Class Library provides mechanisms to support the displays of time-ordered information in a graphical manner. The generic Delphi timeline contains a two-dimensional, composite region displaying resources and their use versus time. Each display is composed of rectangular subregions that represent resources aligned down the vertical axis to be viewed over time, and time across the horizontal axis. Scrollable windows are utilized to allow the user to manipulate the displays to view the required resource profiles. Events for any period of time are modeled and displayed. Each event is indicated on the display with a text-label, color and state information, all of which can be customized for the specific application. The Timeline is a powerful visualization device allowing the user to navigate hundreds of resources profiles. Each event on the timeline can be compressed, expanded, edited, unallocated from its resource, locked and unlocked. Visually the user sees more detailed information the shorter the timespan being viewed, thus allowing a zoom cycle capability for all Timeline displays. Use of interactive point-and-click devices speed the user through birds-eye views of schedules down to the component parts and resources allocated to a single user request on the timeline. The timespan is adjustable and has no technical limits (although practical limits may insist only a portion of large schedules be viewed at any time).

The Timeline Class Library provides class implementations for drag-and-drop features between subregions, event manipulation, and supports multiple views of events. The use of color to convey importance, hierarchy, or groupings can be defined by the Customer, and color changing functions are provided with the Timeline classes. Scroll bars and multiple windows are supported. A time probe that anchors the center of focus of the display and provides default values of the time window of interest to other Delphi components, such as schedule Activity Editors, is also provided. The Timeline classes establish a generic structure for events.

Planning Class Library

The Planning Class Library provides the display and integration structure to all Delphi scheduling applications. This library provides the interface slots to allow a developer to build up from the foundation of the components of Delphi into a coordinated scheduling environment. Additional capabilities provided within the Planning Class Library are: error message display and logging, distribution of resource states and coordination of schedule modifications, resource editing, scheduling activity creation, deletion, and dissemination, (planning synchronization across multiple users).

4.6.5.4 Use Of Delphi In The Planning Workbench

4.6.5.4.1 Customization of the Resource Model Hierarchy

The Planning design takes advantage of Delphi by utilizing the object-oriented mechanisms of inheritance. Elements of the Resource Model hierarchy have been customized for the Planning problem domain. Derived resources have been designed to customize resource state and override or add specialized algorithms and constraints. Some derived resources add scheduling operations which take activities and check constraints and generate resource state through time.

The derived Planning resources have been designed in a parallel hierarchy, as a result keeping the natural resource hierarchy and the scheduling hierarchy separate. Derived activity classes have been created that model the types of activities appropriate for the Planning resource model. A key point is that the specific additions to Planning system have been built upon the foundation already provided by the existing Delphi Resource Model structure. Many additions are simple derivations of existing planning and scheduling objects.

Figure 4.6-1 shows the Planning extensions to the Delphi classes previously described in section 4.6.5.4.4. It can be seen from the diagram that the Planning-specific class `PlResource` has been derived from the Delphi class `RResource`, and as a result, the PlResource class inherits all of the associations RResource has with other classes in the Delphi Resource Model.

4.6.5.4.2 Customization of the Scheduling Class Hierarchy

The Delphi Scheduling Class Library provides a framework for incorporating a variety of algorithms in the Planning Workbench. The Planning Workbench reuses the classes the Scheduling which define protocols for sorting, filtering, resource activity generation, and refinement. It is the goal of the Planning Workbench to employ the Scheduling framework to construct algorithms which optimally allocate activities to resources, especially local disk storage.

Currently, the processing COTS, AutoSys, does not directly track required disk usage when releasing PGEs for execution. Therefore, it is the responsibility of the Planning Workbench to analyze interdependence of PGEs and their file requirements and present the PGEs to AutoSys for

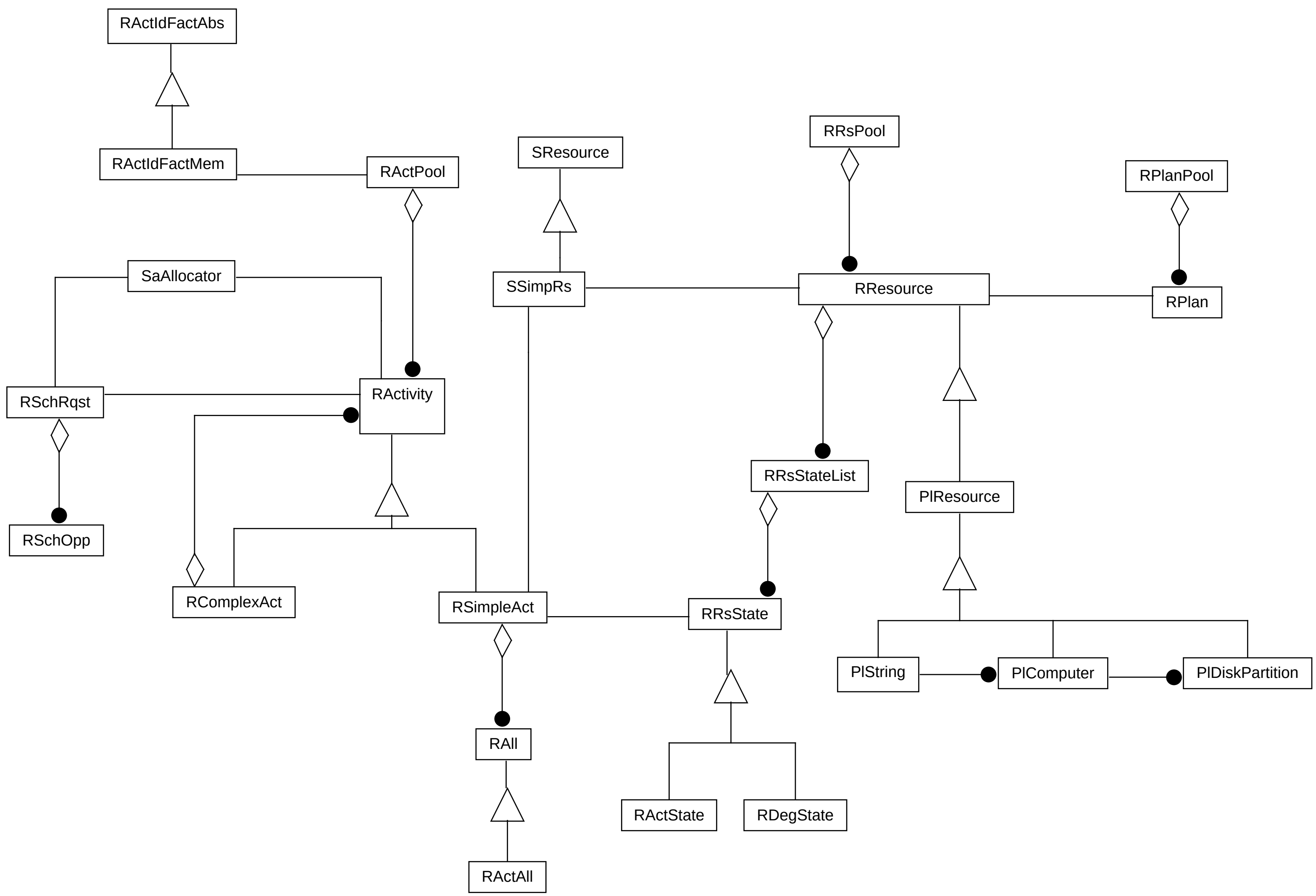


Figure 4.6-1. Delphi Resource Model

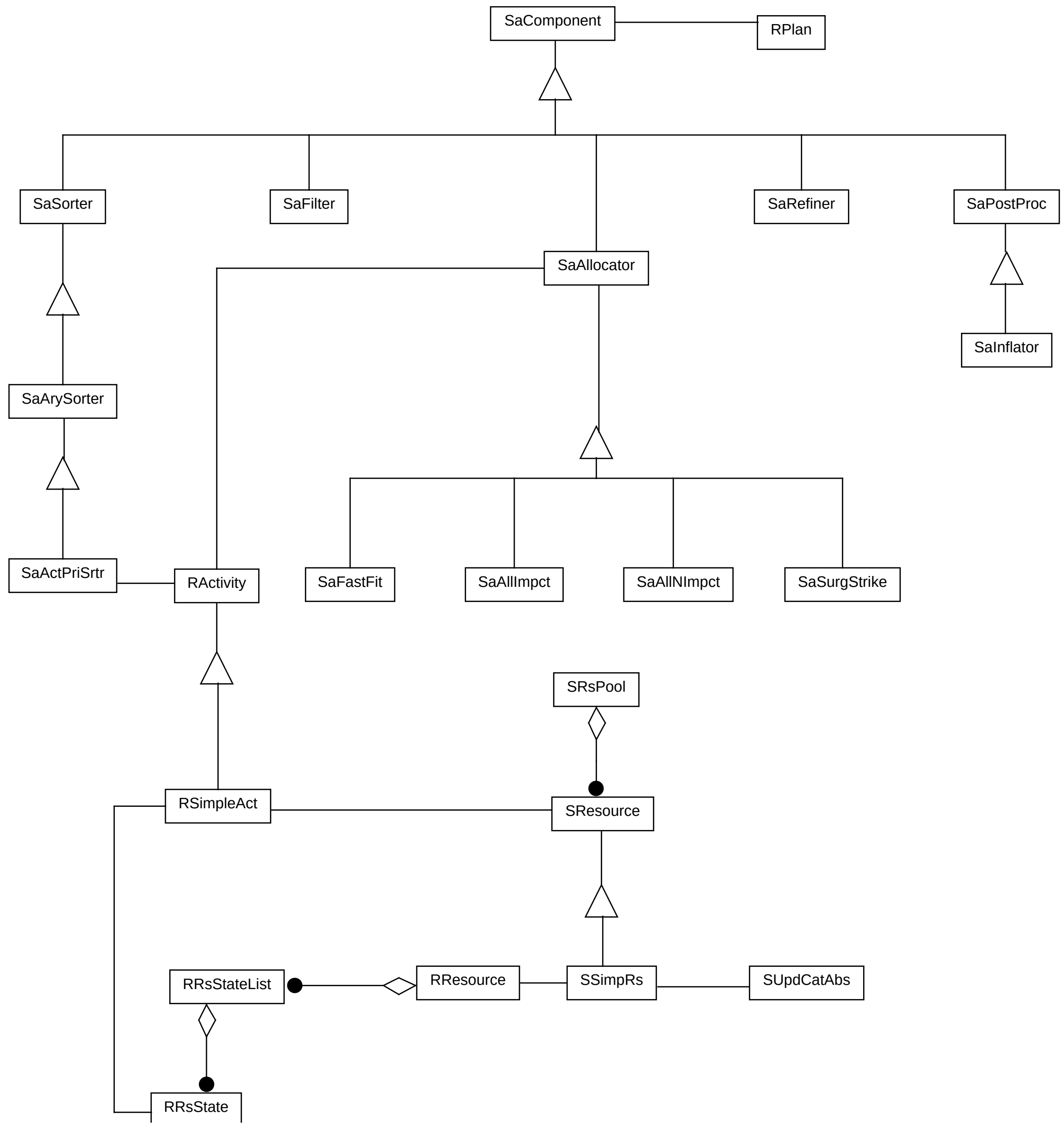


Figure 4.6-2. Delphi Scheduler Model

execution in a way which will maximally utilize disk resources and minimize file swapping between the Data Server and local storage.

4.6.5.4.3 Customization of the Timeline Class Hierarchy

The Timeline is one of the primary visualization tools within the Planning Workbench. The Timeline will provide a GANTT style view of the requested allocation of PGEs. In the Timeline, horizontal boxes will represent PGEs, and the boundaries of the boxes will represent start and end times. The Timeline enables the user to view in an instant the breadth of system resource utilization over time. Information displayed on the box (textual, color, graphical, etc.) will aid the user to search for individual PGEs. Several instances of the Timeline can be viewed simultaneously to compare the results of various scheduling algorithms.

4.6.6 PDPS Database CSC

The Planning and Data Processing subsystems (PDPS) require that much of its data be stored in a repository in order for data processing and planning activities to be executed, managed. and monitored. These data attributes are known as “persistent” attributes, as they need to be housed in a repository “persistently” over time. This repository is a relational database management system (RDMS).

Objects and associations identified in the OMT model are manifested in this repository in one of two main ways. First, objects with persistent attributes are manifested as tables - with columns allocated for the storage of these attributes and with keys that make explicit the implicit relationships in the model. Second, some OMT methods in the identified model become “stored procedures” or “triggers” within the database which, combined, will manage the events which take place within and between objects as plans are made, executed, and managed.

The creation of the PDPS RDMS requires that the OMT model, which is a conceptual or “logical” model of the subsystem objects and functions, be mapped to a real-world or “physical” implementation of these objects and functions.

4.6.6.1 Database Logical Design

The Logical Model of the PDPS, which uses OMT methodologies, is the model of record. This, OMT model, however, is wholly logical. That is, this model can not be the engine for implementation level work as there is no direct connection between the object-oriented concepts of classes, association, methods, and the like to a RDMS. Thus, a “physical” or real-world design equivalent of the OMT must be articulated.

The physical design equivalent for OMT for a RDMS implementation is an Entity-Relationship Diagram (ERD), wherein the design objects have direct physical analogues within the implemented database.

Persistent objects/attributes required by the PDPS subsystem are stored as Sybase tables and columns in the RDMS. To map the OMT persistent attributes to these tables and columns we use an OMT-ER matrix. This matrix facilitates the mapping of a logical model of OMT objects to a physical model of relational database tables via an Entity-Relationship Diagram (ERD).

OMT objects and their associated attributes are catalogued for all objects with persistent attribute requirements. These catalogued objects are then mapped to T-SQL schema level tables and associated columns via the OMT-ER matrix. The resultant matrices from this mapping are listed in the SDPS Schema definition. This catalogue of the mapping of the OMT model to an ERD model is used both for traceability i.e. how objects were “physically” implemented in a RDMS and for tracking derived objects and tables that are identified in the process of this mapping.

The following **logical rules** are used during this mapping of OMT to ERD's

- Rule 1 Objects without persistent attribute requirements are not mapped to the ERD
- Rule 2 Objects with persistent attribute requirements are mapped to Sybase Tables
- Rule 3 Attributes associated with these objects are mapped to columns within these tables
- Rule 4 These tables are keyed with appropriate primary keys.
- Rule 5 Any derived objects, like primary keys, are annotated.
- Rule 6 Any attributes which are at odds with a RDMS implementation (e.g. arrays, lists) are exploded into to derived tables as appropriate. These derived tables are annotated.
- Rule 7 Associations in the OMT model are implemented via primary & foreign key logic

The following **sequenced steps** are taken during this mapping of OMT to ERD's

- Step 1 Objects are mapped to tables
- Step 2 Attributes are mapped to columns
- Step 3 Associations are mapped to primary/foreign key logic
- Step 4 Attributes are annotated with default values, null processing logic, example records and so on.

4.6.6.2 Transformation of the ERD to T-SQL - Physical Design

The identified matrices of the mapping of OMT objects to Sybase tables which are the net effect of the mapping of the OMT model to an ERD are entered into an appropriate ERD CASE product. All derived attributes and tables will entered into this CASE product as well. Rules and steps for entering this information do not apply at this phase as all rules and steps are applied during the mapping. Rather, this is an information transference from text into a vehicle which will produce low level Data Definition Language (DDL) in Transact SQL - e.g. create/drop table, create indexes, and create primary/foreign key relationships.

4.6.6.3 The Design of Normalized Tables

Normalized RDMS tables are tables that meet specified database design criteria and escalating levels of design constraint. At zero normal form all data resides in a single repository - there are zero constraints on how data is stored. Each level of normalization adds a constraint to information storage. For the PDPS database the normalization rules in use will constrain the data to a third-normal form level of design.

This normalization method constrains the data tables in PDPS to meet the following three criteria:

1st Form:Any populated column in a table has a discrete value.

Result:A column in a table that is used to store information about input data can store information about one input data item or a pointer to a list of input items somewhere else in the database but it cannot store a list of input data information within a column

2nd FormAll tables are keyed.

Result:Any record in a table can be fully identified by the column(s) that are the record level identifier.

3rd Form:Each column within the table that is not the primary key or part of the primary key is fully dependent on the primary key.

Result:There can be no partial or transitive dependencies for any non-key columns within a table

All non-key columns in the table are a complete function of the primary key.

4.6.6.4 Database Interface

The PDPS DBMS CSC includes interface classes will be responsible for mapping persistent data in object oriented applications to the Relational Database Management System.

For each persistent class in the object model a corresponding database interface class is required. The database interface classes will be used to retrieve object information from the database, manipulate it, and store it back into the relational database. Each of these classes will have its own set of methods that define its interface to the database (see Figure 4.6-3 for a diagrammatic view, Figure 4.6-4 for a OMT depiction). Typically, each class provides the creation of persistent object instances, updates to attributes, querying using ANSI SQL or stored procedures, and deletion of objects from the database. Each of these methods make the appropriate database calls to implement its functionality in the database.

This interface also manages the connection between the client and server, creation of multiple communication channels for parallel processing, interchangeable submission of statements between the client and server, associated answer and result sets, and exception and error handling. This interface consist of compiled functions on the client software.

PDPS plans to use third party software such as Rogue Wave's DBtools.h or a similar product to ease the development and increase productivity for developing the database interface. The third party product will serve as a wrapper for the SQL functions within the C++ applications.

4.7 PLANG CSCI Management and Operation

The following sections discuss the management and operation of the Planning CI. Specifically, the manner in which the CI is managed at the local and system levels is discussed. The approach to the development of operator interfaces for the Planning CI is then described. Finally, the approach to reporting for the Planning CI is described.

Before examining management and operation of the Planning CI at the system level, the local level management view of this CI is considered. That is, the operations and management of the CI with short range considerations. These include the subscription submittal activities, the production request editor, and the AI&T-developed PGE profile editor. These CSCs are used to create or update data structures local to the planning process. With longer range implications are the subscription manager, and the planning workbench.

4.7.1 PLANG CSCI Operation

A key consideration to remember is that operations personnel will not be required to routinely insert individual data processing requests as has been done in the past for some systems. That is,

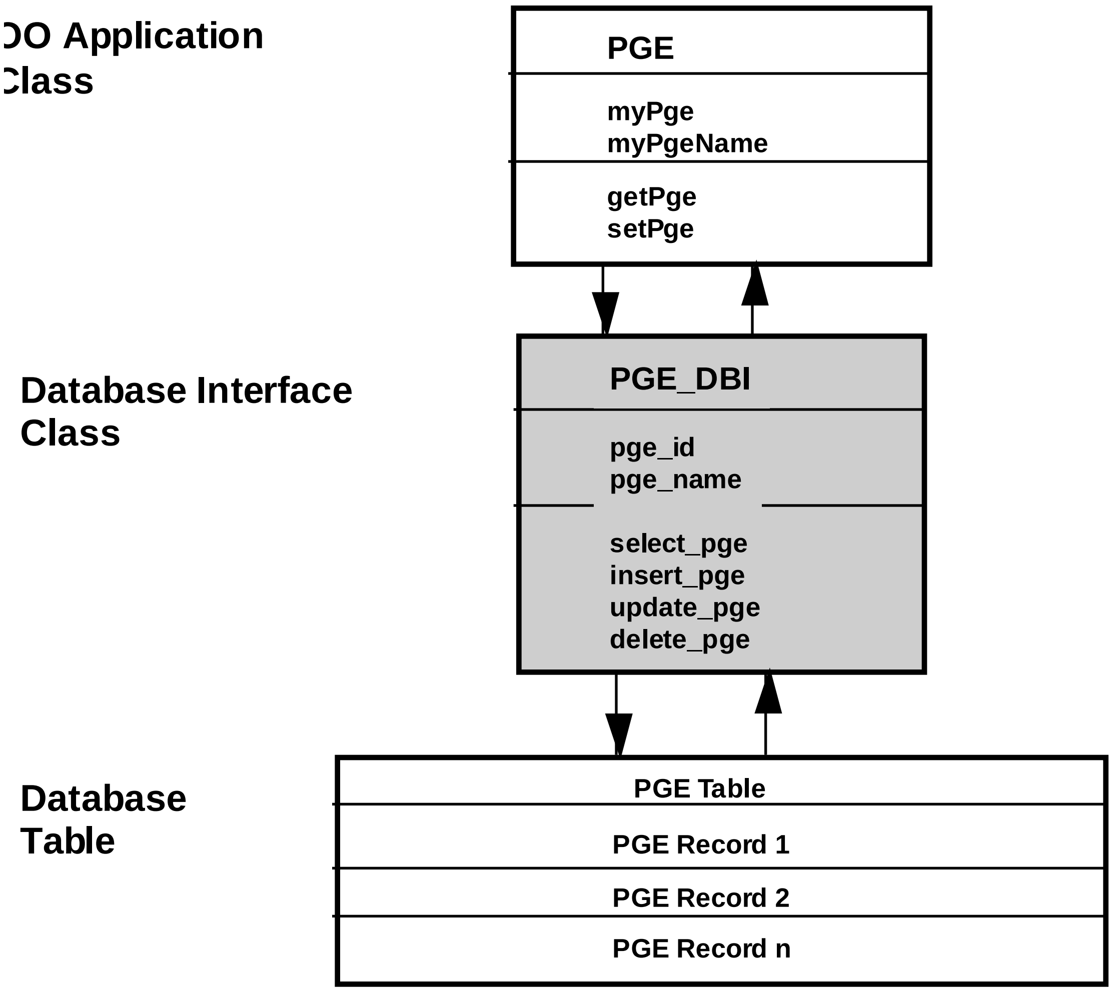


Figure 4.6-3. Database Interface Classes

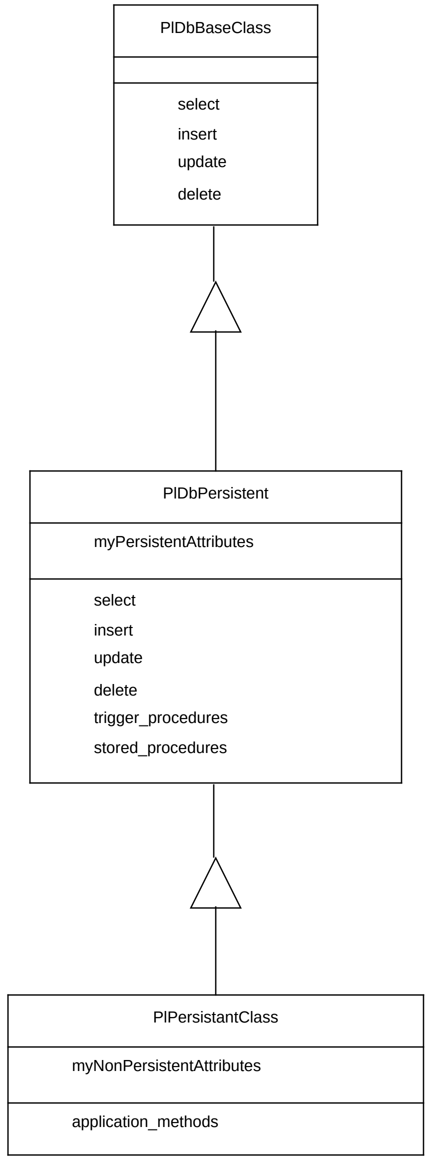


Figure 4.6-4. Database Interface Object Model

once the instructions for processing a type of data with a particular PGE have been inserted into the Planning database, the error-prone, repetitive entry of data processing instructions for each data product is avoided. Data processing will be primarily data driven. Plans are constructed based upon the data that is expected to arrive from external sources or based upon the intent to reprocess existing data with revised algorithms. As data arrives at the DAAC, the PDPS systems will be made aware via subscriptions of the existence of the data and processing will begin when all necessary data files have arrived and are staged.

Production Request Editor

Typical of local management activities is the Production Request Editor CSC which provides the capability to define and modify PRs that are used as a part of the planning process. This interface is used by operations personnel to enter the production requests that is the starting point of the planning process. The PRs do not need to be added/created for each job to be run, but only for each job type or class to be run. The actual job stream or script is built up from this using information contained in the planning database, and submitted as a part of the planning and processing activity.

Subscription Submittal

The Subscription Submittal CSC permits the operations personnel to efficiently enter subscriptions to Data Server/Ingest for one of the types of data needed by a PGE for processing. This CSC is a non-persistent or utility type application that is typically used during the algorithm integration and test phase for the PGE. At that time, the planning database is initialized with the information needed to plan and schedule the execution of the PGE during normal operations. After the subscription has been entered, no further operator interaction is ordinarily required to insure that the subscription notifications are transmitted to the Planning CI. This CSC allows operations to view current outstanding subscriptions, and to modify or delete them as required.

Planning Workbench

The Planning Workbench CSC includes a collection of utilities to enable the operations personnel to perform the planning function for the DAAC. These include capabilities to create candidate plans, activate a candidate plan, update and cancel an active plan. The Planning Workbench CSC allows operations personnel to devise plans for routine production data processing and reprocessing. The production scheduling personnel selects Production Requests from the Planning database and includes them into the planning process. Candidate plans are developed based on the predicted availability of the necessary input data, PGE processing characteristics, planned availability of resources at the DAAC, and priorities and policies established by the local DAAC management. The Planning Workbench CSC allows the production scheduling personnel to create multiple candidate plans for evaluation, select the candidate plan that conforms best with the DAAC system operations plans, and to retain the plan for future modification. The Planning Workbench provides the capability to activate the current plan which thereafter guides the activities of Processing. This CSC also includes the capabilities to update or modify an existing active plan or to cancel an active plan.

A concept of the Planning Workbench CSC is to take advantage of the human capabilities to distinguish a 'good' plan from a less optimal plan. The capabilities provided here allow the operations personnel to generate multiple candidate plans for comparison and evaluations. The operations personnel can devise these plans by varying certain system planning strategy parameters such as priority allowing the planning manager to automatically construct several plans and select from among them the one that is most suitable for the needs of that DAAC.

The key feature here is that each DAAC may construct the planning procedures that most readily fit with existing operations procedures. For example, a planning cycle operations concept developed earlier suggested that a 30 day and a 10 day candidate plan could be prepared by the DAAC operations personnel to establish basic long and short term plans for processing objectives. The Active Plan is assumed to be prepared from the latest 24-hours of the 10 day plan. However, this concept of operations is only one possibility. The operations personnel at each DAAC can prepare the processing plans that meet their needs best, e.g., a sequence of one week plans might be desirable for another DAAC. In addition, the operations personnel may easily adjust the priorities associated with a type of processing (standard, reprocessing, or, for Release B, ad hoc processing) or with individual products. This capability allows the operations personnel to quickly respond to changing priorities determined through NASA procedures.

During the planning activity, the Planning CI attempts to allocate data processing jobs to one of the several processors that it has knowledge of as being available for processing. The Planning CI may allocate a job to specific processor or to one of a pool of processors, depending on the needs of the particular algorithm. In general, a pool of processors is available for data processing. However, processors may be removed from service occasionally for a maintenance or upgrade operation referred to as a “ground event”. In such a case, the Resource Manager for the DAAC will identify to the Planning CI those processors that must be removed from service and the specific time period when they are needed for service. Alternatively, the Resource Manager may identify a resource, a time interval when the resource is required ('next Tuesday afternoon') and the duration of the resource maintenance activity. In the case of the flexible planning window, the Planning CI will select specific time period that best meets the needs of processing. Once the Planning CI has selected the time period for the ground event, that interval reservation for the resource is fixed with respect to any subsequent replanning.

Subscription Management

The Subscription Manager CSC manages the interface with Data Server/Ingest for the receipt of subscription notifications. These notifications are provided to Planning as a part of the process leading to the activation of PGEs. The notifications indicate to Planning that a granule of data needed by a PGE has become available at Data Server/Ingest, for example, as the result of a Level 0 data set arriving from TSDIS. When all data required for processing has arrived, the Planning CI will release the PGE to the Processing CI for activation.

This approach employs the common subscription notification scheme. It automates the process of communicating with Data Server/Ingest CI for potential data arrivals. The Subscription Management CSC is persistent - it is continually active and awaiting notifications. It isolates the interfaces to Data Server/Ingest to a single element for receipt of notifications, simplifying the management of the system. Predictions on the arrival time of required data are generated by Planning. These are used to identify delayed data arrivals, allowing the operations personnel to react to and resolve the situation.

4.7.2 System Management Strategy

The system management strategy as revealed by the PLANG CI is discussed in the following paragraphs.

4.7.2.1 System Management and Operations Philosophy

The primary design consideration for the Planning CI from the point of view of system management and operations is that operator interactions with Planning be simplified and automated as much as possible. In addition, the Planning CI provides operations with the needed flexibility to respond to unexpected tasks as they arise. In part, the objectives of simplification and flexibility are accomplished through well designed GUIs that aid in developing the needed timelines.

4.7.2.2 PLANG CI and the System

The Planning CI is a part of the ECS System. It is integrated with the other ECS components via the common management interfaces to participate in the ECS enterprise. The following paragraphs address certain system management topics from the point of view of the Planning CI.

Startup & Shutdown

The Planning CI is integrated with the system management interfaces for startup and shutdown. Via these interfaces, the operations personnel will control the initialization of the Planning CI including planning database initialization. The controlled startup and shutdown insures that the necessary coordination and synchronization of activities, within the Planning CI and with other subsystems, is enforced. Adhering to these procedures insures the highest availability and reliability for this critical component of the ECS system.

Fault Handling and Recovery

The Planning CI is integrated with the system management components to address fault handling. By providing these fault messages to the appropriate operations personnel via a unified interface, the consistent handling of component and system faults may be addressed. This permits the personnel responsible for address faults to inspect the information at hand from across the system and to aid him/her in identifying the appropriate cause of the fault. This scheme insures that faults are addressed quickly and accurately, enabling the planning and processing of science data to proceed without unnecessarily long delays.

System Maintenance Operations and Ground Events

Of particular importance to the Planning CI from the point of view of system management is the scheduling and coordinating of ground events (e.g., maintenance activities, software upgrades, or for testing activities) with processing. The data processing resources are, from the point of view of system management, another resource that will occasionally be required for ground events. From the Planning CI point of view, these resources are typically dedicated to the science data processing activity and are therefore mostly reserved for planning and processing. The Planning CI supports these views of the data processing resources through the planning process. The system Resource Manager will identify (usually well in advance) the ground event that requires the resources that are typically dedicated to the planning and processing activities. The Resource Manager provides these requirements to the Planning CI for incorporation with its planning activities. These ground events are dealt with by Planning as high priority activities whose resource requirements are to be satisfied. After these events are planned for, the time allocations are locked-in, not to be changed by subsequent planning and scheduling actions.

In a similar manner, as new hardware resources are introduced into the system to support science data processing, these resources are identified to the Planning CI as potential data processing platforms but which are indefinitely scheduled out-of-service until the resources are fully integrated and tested and released for use by processing. During the I&T phase for these new platforms, they are scheduled accordingly. In this way, the Planning CI cooperates in the system-level management of the data processing resources.

4.7.2.3 Interfaces

The interfaces of the PLANG CI and the support and coordination required for system management and operations is discussed in the following paragraphs.

MSS

The Planning subsystem interfaces with the MSS for management related activities. As discussed above for startup & shutdown, fault handling, and resource planning, the PLANG CI participates with the MSS in the enterprise management activities to insure that Planning supports the system objectives in a seamless fashion.

Data Server/Ingest

The Planning subsystem interacts with the Data Server/Ingest via the subscription interface to achieve the system objectives of responsive and efficient data processing. The subscription notification scheme for interfacing with Data Server/Ingest is a very flexible and robust approach for signaling the presence of data to be processed. It removes operations effectively from the loop of coordinating transfers of data. This simplification greatly improves the reliability of an interface requiring operator interactions. In addition, the PLANG CI interface with the Data Server/Ingest for staging of data prior to processing is an automated approach that, again, removes operations from the roll of handling data transfers thereby making more reliable and timely the movement of data for data processing.

Processing

The major interfaces with the PLANG CI are to the Data Processing subsystem's PRONG CI which are in most cases implemented through the planning database. Interfacing through the database provides a simple and reliable scheme for the transfer of schedule information from one CI to the other. The separation between the planning activity and the processing management activity insures that as the requirements and technology evolve on either side of the separation, the impacts to the other side of the interface can be controlled and limited.

AI&T

The AI&T activities in relation to the Planning activity are limited. AI&T activities populate portions of the Planning database as a part of the installation of the science software at the DAAC. This pre-operational activity provides for the people intensive effort of understanding the planning and processing needs of the science software and then imbedding that extracted information into the planning databases.

4.7.3 Operator Interfaces

This subsection describes the operator user interfaces provided by the Production Planning CSCI to operations personnel. A general description of the framework and methodology employed for the development of these interfaces can be found in section 4.5 of the detailed design overview

(305-CD-001-001). This subsection augments that information with additional design information which is specific to the Production Planning CSCI.

The operator user interfaces will be developed from a combination of COTS graphical user interface (GUI) libraries and custom code. Hughes Delphi Scheduling Library will provide some components for reuse, such as the Timeline, but these will have to be customized with Production Planning code. In addition, custom graphical interfaces will be created with the aid of Integrated Computer Solutions' Builder Xcessory™. Builder Xcessory enables the developer to manage Motif graphical user interface projects by providing a WYSIWYG, drag and drop, visual develop environment. Once an interface is constructed, Builder Xcessory will generate C++ code which represents the GUI and encapsulates the C-based Motif Widget set. The generated C++ code can then be combined with other Production Planning specific code.

4.7.3.1 Off-The-Shelf Interfaces

As discussed in section 4.5.5, the Production Planning CSCI will reuse part of the Hughes Delphi Class Library to develop graphical user interfaces for the Production Planning subsystem.

4.7.3.2 Production Planning User Interfaces

This section is intended primarily to describe the data that may be displayed to facilitate the Operations of the Planning utilities. The precise determination of the GUI will be decided by surveying the wishes of the users of the interfaces and demonstrating prototypes. The discussion below does describe some interaction between the user and interface, this discussion is only preliminary and used to illustrate one of many possible arrangements.

Production Request Editor

The Production Request Editor is a utility which provides operations personnel with the capability to define and modify Production Requests (PRs) that are used as part of the planning process. The GUI for the Production Request Editor will be constructed with the aid of Builder Xcessory and combined with custom code which interfaces to the Planning database for storage of PRs.

Upon invoking the Production Request Editor, the operator may be presented with a main window which serves to create new PRs and display stored PRs. The information to be displayed consists of two parts:

1) A product selection sub-section for choosing a unique product
   - Any of the information associated with the products that is stored within the PDPS database may be displayed to facilitate the selection. This information includes Product Name, Product Version, Instrument, Platform for the product. A table format may be suitable for this information. The default PGE that generates a selected product may also be displayed.

2) A Production Request sub-section for defining the PR to generate the selected product.

   - A production period which consists of starting and ending dates and times is required. In addition, an editable text field displays the priority of the PR. The operator can accept the default priority for the PR or enter a new value. Once the PR production period and priority are specified, the operator is encouraged to enter a comment to aid in uniquely identifying the PR for later recall.

If the operator chooses, he or she can override the default PGE by selecting. The operator could be presented with a new window which provides a list of PGEs corresponding to the selected instrument and product type.

The user to invoke a dialog for overriding specific parameters for the chosen PGE. A dialog can present the user with a list of parameters and values for modification.

The utility provides a means of reviewing, editing, and deleting previously defined PRs.

The second set of functions available from the Production Request Editor is to review / modify the Data Processing Request which fulfill the Production Request. This interface is being prototyped within the IR-1 development and may be used to illustrate the data that can be diplayed.

Ground Event Editor

The Ground Event Editor provides a means of defining ground events to the Production Planning database. The GUI for the Ground Event Editor consists primarily of a window for entering and editing ground events. When the operator first invokes the Ground Event Editor, he or she can begin to enter new information into the provided fields or recall a previously defined ground events for editing.

Defining a new ground event is a two step process. The operator first provides information to identify the ground event and then chooses the resource on which to schedule the event.

When identifying a ground event, the operator begins by entering the name of the ground event. The operator then enters the priority of the ground event or accepts the provided default value. Time and date fields are supplied for entering the earliest start time and the latest end time for the event. The operator can then toggle the button which indicates that the ground event is to function as a template. If the ground event is a template event, it will remain in the database indefinitely and can be used to quickly define new ground events. Next, the operator can enter comments to describe the ground event for ease of recall and for use in reports.

The next step in creating a new ground event is to select the resource on which the event is to be scheduled. The operator is presented with a list of available resources from which to choose.

Once the ground event information is entered, it can be saved to the database or canceled. The operator can also use the Ground Event Editor to recall previously defined ground events for modification or deletion.

Production Planning Workbench

The GUI for the Production Planning Workbench includes windows for creating plans and a timeline for displaying plans. When the operator invokes the application, a window appears which allows the user to specify the name of a new plan or select a plan which was previously created. To create a new plan, a window is provided to define the plan parameters. The operator first enters the plan name and then enters the starting and ending dates and times. In addition the operator can enter comments to describe the plan.

After the plan parameters are initially defined, a window may be presented which allows the operator to select the Production Requests to be included in the plan. This window could include two lists: a list initially full of unscheduled Production Requests and a list initially empty of scheduled Production Requests. The operator can choose Production Requests and use arrow buttons to move the selected entries back and forth between the scheduled and unscheduled list.

A timeline window is available to display scheduled events in a graphical format. As described in previous sections, the timeline is built on the hierarchy established in the Hughes Delphi Timeline Class Library and is tailored to Production Planning needs. The timeline is a GANTT chart which displays resources and their use versus time. When the timeline appears, it is first populated by ground events which have previously been defined by the Ground Event Editor. As the operator schedules Production Requests for a plan, the corresponding DPRs appear in the timeline window. If the operator unschedules a Production Request, the corresponding DPRs will be removed from the timeline. See section 4.6.5 for a further description of the timeline.

Several instances of the Production Workbench can be invoked to compare the results of different plans in order to select the best plan to submit for production.

Subscription Request Editor

The Subscription Request Editor contains a single window from which an operator can enter product subscriptions to the Data Server. Two lists could be provided: one list containing unsubscribed products and the second list contains subscribed products. With the use of arrow buttons, the operator would be able to select and move products between the subscribed and unsubscribed product lists.

4.7.4 Reports

A variety of ad-hoc and canned reports will be available to the DAAC operations staff to assist in monitoring of the activities associated with the Planning CSCI. These reports are readily accessible given that Planning CSCI persistent data is maintained in the PDPS Database, a SYBASE RDBMS. Also, ECS application management information is maintained in the MSS database, which is used to log system events. The canned reports will include the following:

1) Candidate Plan Characteristics Report- This report will be associated with each generated candidate plan and will contain summary information to be used to establish the quality of the produced candidate plan. Information contained in this report will include the following:

    a) Definitions of inputs, i.e. Resources, Production Requests, and Priority Information, and other planning data used to create the candidate plan.

    b) Candidate Plan planned results. This will include the number of science data products produced, number of DPRs executed, estimated time to meet all planned activities, estimated resource utilization.

2) Production and Data Processing Request Status Reports - These reports will indicate the status associated with a Production Request and its associated Data Processing Requests. This information would indicate whether a given Production Request is active or inactive in the current activated Production Plan. If the Production Request is active, the last reported job status for the associated Data Processing Requests would be provided.

3) Planning Workload and Processing Turn-Around Reports - These reports will provide tracking information on planned vs. actual processing results. The information provided will include job statistics for a Data Processing Request to allow comparisons in planned vs. actual resource consumption, planned start and end time vs. actual start and end time, planned resource, i.e machine, allocation vs. actual resource, i.e. machine, allocation, etc.

4) Planning Management Reports - These reports will provide the operations staff information on Planning application software events which have occurred. This information will be available from the MSS database.

Other ad-hoc reports can be defined to assist the Production Planning Operations staff in performing their activities. The PDPS Database is the repository used to maintain information on Production Requests and associated Data Processing Requests, Data Subscriptions, PGE Profiles, etc. These reports can be used to track the modifications and provide historical information on these data objects. Because of the use of a consistent RDBMS throughout ECS, the sharing of information between different databases is simplified and will allow for consistent definitions for any number of reports.

This page intentionally left blank.

5. Planning Subsystem Hardware CI

The Planning Subsystem includes a single Hardware Configuration Item (HWCI) which provides the hardware resources to support its production planning and plan management for data processing production management (plan implementation) functions. The Planning hardware consists of one or more Production Planning/Management workstation(s) and a Planning (DBMS) server.

5.1 Hardware Design Drivers

The Production Planning/Management server(s) will support the Planning operations staff in performing their routine production planning and management functions. Workstations are provided for operations personnel access to management GUIs. These functions include candidate plan generation, plan activation, entry of production request information and report generation. The server and workstation classes chosen will be based on projected planning workloads for that DAAC.

At the heart of the Planning subsystem is the planning and production database, with the database server(s) providing the persistent storage for data which are shared between the applications. The database marshals requests for concurrent access to data and provides the protocols to allow applications to be allocated to distributed platforms.

Design decisions have been made since PDR that affect the architecture of the Planning and the Data Processing Subsystems. A primary design driver is the need to maximize the use of COTS provided capabilities and to minimize custom code development. Some design elements that were previously mapped to Planning have been moved to the Processing CSCI. The underlying catalyst for this change has been the selection of the COTS products, *AutoSys* and *AutoXpert*, which will be integrated into the planning and production subsystems providing a basis for monitoring and management of ECS' science data production facility. The Production Management which has been mapped to Planning has been divided between the Planning and Processing CSCIs. Production Management, consisting of:

- managing subscription notifications from the data server and ingest, and
- managing active plan by receiving status feedback from Processing,

has now been divided between Planning and Processing. The net result is a software architecture with lower internal coupling. Planning and production management shares a common database (i.e. Sybase RDBMS) that eliminates the large amount of common data structures existing in the preliminary design. These design decisions have driven the hardware architecture and will continue to do so throughout the prototype analysis which extends through CDR.

5.1.1 Key Trade-Off Studies and Prototypes

Most of the analysis with respect to planning and design investigations to date has focused on COTS suitability and subsequent design partitioning ramifications. Since these aspects are of primary importance to the Planning Subsystem design on the whole the studies and prototypes have not focused on hardware issues alone, but examine the planning design options and requirements as an integrated system. Hardware support requirements are derived in this manner for this subsystem.

PRODUCTION MANAGEMENT COTS The exact nature of the COTS integration is currently under investigation. A number of prototyping activities have been initiated to optimize integration of AutoSys and AutoXpert on selected platforms and ascertain the preferred method given the unique requirements of monitoring and managing ECS' science data production facility. Results of these prototyping efforts will continue to impact the partitioning of functionality between Planning subsystem hardware and the Data Processing Subsystem (i.e. Queuing) hardware.

AutoSys usually comes bundled with Sybase or Oracle, and supports direct query/command access to the database. A limited C language API structure implementation exists. "Job Boxes" allow grouping of smaller, related jobs. Both data-availability ("file watcher") and inter-job dependencies are handled easily. Reporting capabilities are limited (e.g. only start and stop time reported). The system has fault tolerance capacity with the ability to run backup or "shadow" processors. (Primary and secondary hardware capacity is discussed in section 5.3 on failover and recovery.)

A simulation of a CERES chain of PGE's was run across three machines, representative of the class of systems *expected* to satisfy planning execution environment requirements:

- Sun SPARC 10,
- HP 735, and
- SGI Indigo.

The following aspects were tested: inter-job dependencies, file-watchers, cross-platform execution, and the usefulness of the console GUI and the alert manager. All features performed as expected. A test of a series of two second sleep jobs was run on one platform to determine where overhead might be present in the system. Although some latency was found to occur in the early stages of job execution, the cost in time was found to be negligible when compared to the total estimated times for a real ECS job.

DATABASE A study of relational and object-relational DBMS' were performed. Results of this study was documented in the technical paper entitled: DBMS Benchmark Report, 430-TP-003-001. As a result of this DBMS study, four DBMS' were selected for further evaluations: Illustra 2.3.1, Objectstore3.1, Sybase SQL Server 10, and UniSQL. Hands on experience were gained with the products and testing product maturity. Tests were conducted on a Sun/Sparc 20 running Solaris 2.3. Three sizes of input data up to 3 million granules were used to test performance scalability. Recent subsequent evaluations have resulted in selection of the Sybase DBMS as the common COTS for the ECS (SDPS & CSMS).

5.1.2 Sizing and Performance Analysis

The discussion on sizing and performance analysis provided here is a synopsis of the process undertaken prior to and for CDR. The DAAC Specific Volumes (for GSFC: DID-CD-305-014, for LaRC: DID-CD-305-015, for MSFC: DID-CD-305-016, and for EDC: DID-CD-305-017) contain full details on site configuration as well as sizing and rationale. The sizing and performance analysis applies primarily to LaRC and MSFC for Release A, due to their requirements for TRMM mission support. (See those volumes for more details.)

The Planning (DBMS) server(s) will support the Planning database that contains all the information central to the functioning of the Planning Subsystem. To size the Planning (DBMS) server(s), it is necessary to estimate the size of the Planning database (for a pre-defined time period) and to determine the database throughput it would have to support.

The estimated size of the Planning database for a pre-defined time period can be determined by developing a model of the Planning database that identifies the type and amount of data to be tracked. The seven principal categories of data to be tracked in the Planning database are as follows:

(1) PGE profiles,

(2) Production Requests,

(3) Data Processing Requests,

(4) Plans,

(5) Ground Events,

(6) Data Granule, and

(7) AutoSys.

DATABASE SIZING Based on the number of activations per day per PGE for a given epoch (as established by the January 1995 ECS Technical baseline which incorporates Ad Hoc Working Group on Production (AHWGP) information), the size of the Planning database (to support one month's worth of processing) can be estimated for a specific instrument. The number of activations per day can be used to determine the number of Data Processing Requests to be tracked. In addition, the number of Production Requests can be assumed to be a function of the number of Product Generation Executives (PGEs) identified (AHWGP baseline supplied).

Assuming one candidate plan generation and two plan activations per day, the estimated size of the Planning database, to support the case of one month's worth of CERES processing during Epoch c, is currently estimated to be around 2.7 Mbytes total for Release A at LaRC.

Initial Planning database size estimates and details on the assumptions used can be found in Planning Subsystem Database Size Estimate for the ECS Project. (Reference: 440-TP-012-001). This analysis has been revised and updated in a spreadsheet analysis in an addendum. Assumptions are provided in the addendum.

Database sizing estimates for Release A (i.e. 2.7 Mbytes) are relatively low. The Planning database server will be scaled for 3Q99, Release B at LaRC by examining Epoch k PGE requirements. MSFC sizing is performed to take into account the entire TRMM LIS mission. The total database size at LaRC for Release B is 31,887 KBytes or 31.9 MB.

DATABASE THROUGHPUT REQUIREMENTS To determine the Planning database throughput, it is necessary to determine the number of times the Planning database would have to be accessed to either retrieve or store a certain amount of data, for a pre-defined workload. The planning daily workload can be divided into the following major activities:

(1) Production request processing,

(2) Plan creation,

(3) Plan activation,

(4) Plan feedback updates, and

(5) Report generation.

For each of the activities listed above, the number of read/write accesses (of 12 Kbytes each) from/ to the Planning database can be estimated based on the aforementioned number of PGE activations per day. This is currently estimated to be on the order of 50,400 database accesses per day (background load) to support CERES processing during Epoch c only. (This equates to be a background load of less than 1 access/second.) The peak load (occurring during plan creation/ activation) is currently estimated to be up to 30,000 database accesses per plan or about 100 accesses/second (assuming a plan takes 5 minutes to generate). Actual estimates of the Planning database throughput and supporting details can be found in the Planning Subsystem Database white paper (Reference: 440-TP-012-001).

The planning task is client/server oriented. Beyond human interaction (via the Planning component), the real specification and sizing rationale is established by transaction rates. Transaction rates can be approximately predicted based on Technical Baseline data, The number of processing activations per instrument at each DAAC site also provides insight into the number of transactions.

Static disk sizing is a significant factor in sizing and takes into account the database sizing and COTS load requirement estimates (and prototyping measures) into account. The provided capacities are given in the DAAC Specific Volumes for the operational sites (LaRC and MSFC).

The Planning Server component would be is expected to require only one small server workstation class machine (due to RMA requirements) for its execution at normal periodic intervals (e.g., one session per day). The Process Queuing server, discussed as part of the Science Processing hardware (SPRHW), acts in cooperation with and serves as a backup for this host.

Since one will act as a backup for the other, two small servers are required due to their role in supporting the production planning and task queuing management functions of the entire DAAC's processing complement. In the event of failure of the primary host hardware, transition to a secondary host will be facilitated. (See discussion on disk mirroring and primary/secondary hosts within Section 5.3 which discusses failover and recovery strategies).

5.1.3 Scalability, Evolvability, Migration to Release B

With respect to Release B migration, Planning is somewhat of a special case. Database sizing and processing were evaluated for Release A (Epoch c), which resulted in relatively insignificant performance and capacity requirements. Since the Release A operations, given the current launch and mission timelines, is relatively short before Release B operations commence, it was recommended that the Release A server platforms be sized to Release B requirements for LaRC. (Note that MSFC platforms are sized for the entire TRMM mission, therefore, Release B is not a concern in this case). As a result, sizing estimation for Release B was undertaken as synopsized above, and this information was used to size the Release A server hardware complement.

With respect to the Planning Subsystem, general growth and evolvability strategies are not complex. The basic strategy, generally applied to client server configurations, is application of technology refresh through "swap-out" of selected components. Due to Planning's importance to the DAACs, this would have to be performed in parallel to DAAC operations, with initial application of new hardware technology (servers, workstations, disk, etc.) applied within the AI&T area (with later transiton to operations support). With respect to scalabilty and evolvability issues outside of general technology refresh, the following growths paths are applicable to the Planning Servers and supported by the current design:

- *Possible Use of SMP Servers* -- Depending upon DAAC specific performance requirements, the configuration could be a primary and secondary CPU within a single server, e.g. SMP. The benefits to this approach would be "rightsizing" of the server platform at a site to meet initial requirements, with an inplace upgrade capability to meet out-year performance requirement growth. The feasibility of an SMP configuration is currently being investigated, with the central issue being software and COTS compatibility with the SMP architecture.
- *Disk Requirement Growth* -- The current Planning Subsystem design allows for both standard and RAID disk application without rework of the core design. This can be accomplished through the application of channel adapted disk capacity upgrades (e.g. larger RAID units), or through network adapted disk servers addressible by the DBMS servers. These growth decisions, which are likely to occur at the DAAC due to science mission growth and/or expansion, will be made through predictions on future DBMS transaction rate needs, as well as I/O and storage requirements.

5.2 HWCI Structure

A block diagram of the Planning hardware is shown in Figure 5-1.

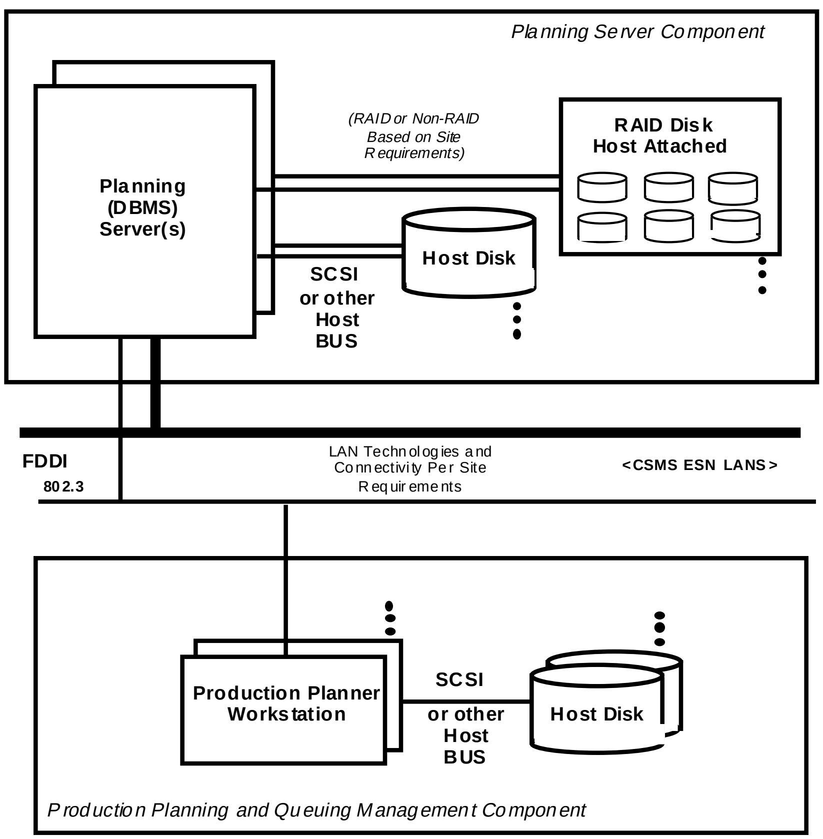


Figure 5-1. Planning Block Diagram

The Planning Subsystem components maintain status and database information on: production plans, ground resources, resource status, resource load, etc. The Planning Subsystem coordinates the Processing Subsystem activities through data processing requests.

5.2.1 Connectivity

The intra- DAAC data interfaces will be implemented as follows: data interfaces will be of the channel type (e.g., SCSI II); Control interfaces will be of the network type (e.g. FDDI, Ethernet).

The Planning and Data Processing subsystem network connectivity is illustrated in Figure 5-2. Both subsystems will connect directly to the same FDDI ring. This figure highlights the primary components of the Planning HWCI (shaded components apply to the Science Processing Subsystem). Hosts (servers and workstations) will contain single-attached station (SAS) cards and will be connected to an FDDI concentrator, which will in turn be connected to the FDDI switch via a physically wired FDDI ring. Refer to Volume 0 for a general description of DAAC networks, and to the DAAC Specific volumes, for MSFC and LaRC, for details on specific CSMS topologies.

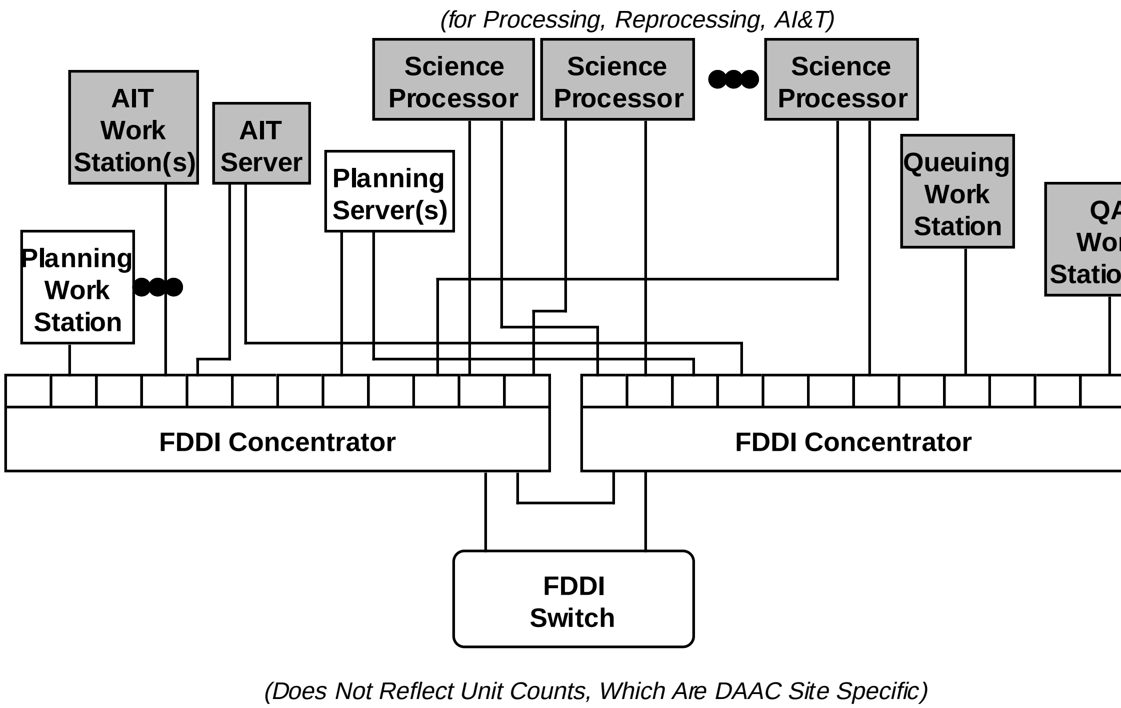


Figure 5-2. Planning Generic Network Connectivity

In Release B, the network connectivity, as far as network types, is unlikely to change. This is due to the performance requirements, so far measured by prototyping and estimated through analysis of planning requirements, do not demonstrate the need for insertion of higher speed technologies.

5.2.2 HWCI Components

This subsystem consists of two hardware components:

- Planning Server
- Production Planning and Queuing Management

Each of these components is discussed in the paragraphs that follow and are elaborated on in Table 5-1.

Planning Server Hardware Component The Planning Server consists of one or more server class machine maintaining status and planning database repositories. Sets of planning servers are provided for each DAAC site whose complement requires science data processing. The relative size and number of these servers is DAAC specific and timeframe specific. For purposes of discussing this design generically, two servers are assumed. One provided by the PLNHW HWCI, acting as the primary DBMS server, and the second provided by the SPRHW HWCI acting as the production queuer and/or secondary DBMS server as necessary.

Production Planning/Management Hardware Component This component contains the hardware necessary to support DAAC operations users staff performing routine production planning and task queuing production management operations functions. This component consists of small workstation(s) (one or more, depending upon DAAC site and system release requirements primarily affected by Operations Staffing needs).

Table 5-1. PLNHW Logical Components and Equipment Classes

Component Name	Class/Type	Comments
Planning Server	DBMS Server (Future & instrument processing specific)	• From small to large server in class, depending upon the site and system release in question. • Predicted for Release-A are small DBMS workstation based servers. • Typically, two are provided per site to assure fail soft operations of the Data Processing Subsystem and the Planning Subsystem. • Interfaces with other subsystems via the ESN LAN interfaces.
	RAID Disk (Host Attached)	• RAID storage for storage of candidate and active plans, as well as DBMS table and working space. • RAID technology not necessarily required for Release-A due to small volumes and the interim nature of TRMM operations at that time. Standard disk may or may not be used for Release-A at some DAAC sites.
Production Planning and Queuing Management	OPS Workstation(s) (small)	• One or more operations workstations, on a site by site basis that support local planning. • Provides workstation processing, I/O and disk resources. • Interfaces with the Planning Server Component via ESN LAN interfaces.

5.3 Failover and Recovery Strategy

The Product Generation function has the following RMA requirements:

(1) Availability: 0.96

(2) Mean Down Time:<4 Hours.

This function is supported by the Planning workstation server jointly with other subsystem hardware including Science Processing's , DBMS/Queuing server workstation, and the science processors as well. The RMA requirement is readily met (reference: Availability Models / Predictions; 515-CD-001-003). However, the DBMS server function is critical and should have additional backup capability.

The basic approach to Planning subsystem failover and recovery strategy for Release A is to provide functionality within the DBMS server for the functional equivalent of RAID disk that provides a mirroring capability. In the case of CPU or power supply failure, a sparing philosophy will be used., which can easily meet the MDT of less than 4 hours

This section describes failure and recovery for the Sybase SQL Server for the Planning and Data Processing Subsystems (PDPS). The Sybase Server will consist of databases for Planning and Database Processing, including AutoSys and AutoExpert. This paper will also be used to determine if additional software is required for failure and recovery and may used to supplement DID 305 failure and recovery material.

This section will address failure and recovery for both system disk crashes and CPU failure.

Planning Server Architecture The PLNHW server architecture is based primaily on providing the facilities with which a common RDBMS for planning and queuing can be serviced. Whether or not this requires one physical server depends upon the RMA requirements, the timeframe of the implementation and the DAAC specific performance required (a function of the routine processing allocated to each site in question). Note that Figure 5-3 assumes an environment with two physical servers. This decision is a DAAC specific one as stressed before.

For purposes of describing the generic PLNHW architecture, it is assumed that there are two physical servers: one being the Planning Server and the second being the Production Queuing Server (actually allocated within the SPRHW HWCI). These servers act as the primary and secondary unit, with the Planning Server acting as primary. Both physical servers act as the DBMS Server. In the event the primary fails the secondary can take over with little or no manual intervention. The secondary assumes the role of the primary and all work is performed using the secondary server until the primary is restored.

Depending upon DAAC specific performance requirements, the configuration could be a primary and secondary CPU within a single server, e.g. SMP. The feasibility of an SMP configuration is currently being investigated.

Disk Mirroring With respect to disk failure recovery, the server(s) will have multiple physical disks to service safe storage of the production DBMS. The disk will be grouped into two logical sets and the second set will mirror the first set. There are two basic strategies that can be employed:

(1) *Hardware supported* -- physical cross strapping of disk between two servers, with one acting as the primary, (hardware supported) and

(2) *Primarily software based* -- disk mirroring between a primary and a secondary server through the use of protocols and LAN networks.

Figure 5-3 depicts a primary and mirrored disk. Given that the RMA requirements, for the Planning and Processing areas, as mentioned above, are not severe, it is likely that the method employed will not be hardware based, but will be based on the second strategy given above. This method, since it is primarily software based (other than the requirements for multiple disk units), is only briefly outlined here.

Sybase mirroring provides a form of redundancy to protect against hardware failure and to provide a degree of fault tolerance. Mirroring is the capability to maintain a replicate of all data stored on a database device. Disk mirroring can provide non-stop recovery in the event of disk failure. A Sybase procedure can be executed to cause a SQL Server database device to be duplicated, i.e. all writes to the device are copied to a separate physical device. If one of the devices fail, the other contains an up-to date copy of all transactions. Mirroring selected devices minimizes disk resources and performance degradation but requires manual intervention to restore the unmirrored devices from backup for hardware failure situations. Mirroring all devices such as the master device, user databases, and transaction logs provides a non-stop recovery from hardware failure, but provides a slight degradation in performance. In the event of media failure, the mirror can take over, typically without any downtime. When the damaged device is repaired or replaced, it is synchronized with the undamaged copy.

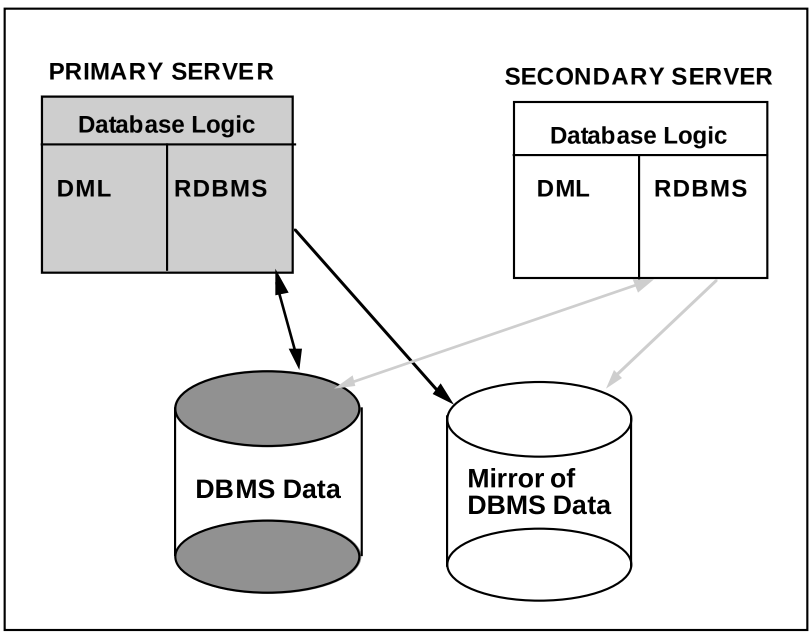


Figure 5-3. Primary and Secondary Server Concept Overview

In figure 5-3, gray areas indicate the primary server and the primary disk. The primary server writes to both the primary disk and the mirrored disk to guarantee continuous operation in the event of a failure. The primary disk is also used for read operations. The secondary server will only access the disks if the primary server fails. If the event that the secondary takes over, it will read and write to the primary disk and write to the mirrored disk.

5.4 Pertinent References

1.. Planning Subsystem Database Size Estimate for the ECS Project, Technical Paper, February 1995, 440-TP-012--12

2. Addendum - Planning Subsystem Database Size Estimate, Spreadsheet Analysis, ***Unpublished***

Appendix A. Requirements Trace

The Interim Release 1 (Ir1) and TRMM Development (Release A) Level 4 requirements listed in the following table reflect the state of the RTM database on July 15, 1995.

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-00005	The PLANG CI shall accept priority Production Requests for the generation of specific Data Products.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue;
S-PLS-00010	The PLANG CI shall accept Production Requests for specific Data Products with associated time windows that are to be routinely generated.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue;
S-PLS-00020	The PLANG CI shall generate Data Processing Requests from Production Requests.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlInputDataType; PlDataGranule;
S-PLS-00040	The PLANG CI shall reject a Production Request if an invalid product identifier has been specified.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue;
S-PLS-00050	The PLANG CI shall reject a Production Request if an unauthorized User Identifier is specified.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue; PlUser;
S-PLS-00060	The PLANG CI shall support the capability to display a response message to the operations staff, indicating the acceptance / rejection status of Production Requests and the reasons for rejection (if applicable).	PlProductionRequest; PlPlanningUserUI;
S-PLS-00180	The PLANG CI shall validate updates (modifications / cancellations) to existing Production Requests.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlInputDataType; PlDataGranule;
S-PLS-00200	The PLANG CI shall accept updates (modifications / cancellations) to Production Requests entered by the operations staff.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlInputDataType; PlDataGranule;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-00220	The PLANG CI shall support the display of a response message to the operations staff, indicating the acceptance /rejection status of updates to a Production Request.	PlPlanningUserUI;
S-PLS-00260	For each Production Request being processed, the PLANG CI shall interact with the appropriate instance of the SDSRV CI to determine whether the Granules needed to satisfy the request exist.	PlPlanningUserUI; PlProductionRequest; PlDataType; DsQuery; DsESDTReferenceCollector; DsESDTReference;
S-PLS-00300	The PLANG CI shall accept ground events to describe the allocation of data processing resources to non-production tasks.	PlPlanningWorkbenchUI; PlResourceManagersUI; PlGroundEvents; PlGroundEvent;
S-PLS-00310	The PLANG CI specification of ground events shall include priorities, dependencies, and estimated duration.	PlPlanningWorkbenchUI; PlResourceManagersUI; PlGroundEvents; PlGroundEvent;
S-PLS-00400	The PLANG CI shall maintain Product Generation Executives (PGEs) information that identifies the Science Software, the order of execution, the conditions for execution, the processing environment, and the input / output data types and locations.	PlPGEProfile; PlPGE; PlResourceRequirements; PlUserParameters; PlDataTypeRequirement; PlDataType; DpPGEProfileEditor;
S-PLS-00410	The PLANG CI shall support the capability to display (via GUI) a list of PGEs maintained in its PGE information database.	PlPGEProfile; DpPGEProfileEditor;
S-PLS-00420	The PLANG CI shall support the capability to browse (via GUI) the information maintained on a specific PGE.	PlPGEProfile; DpPGEProfileEditor;
S-PLS-00430	The PLANG CI shall support the capability to (a) allow (authorized) operations staff updates (enter / modify / delete) of PGE information in the Planning PGE information database, (b) maintain a record of updates made.	PlPGEProfile; DpPGEProfileEditor;
S-PLS-00440	The PLANG CI shall maintain Production Rules that define the production strategy (rules defining production priorities and preferences) to be used when preparing a Production Plan.	PlProductionRequest; PlDPR;
S-PLS-00450	The PLANG CI shall support the capability that allows the operations staff to update (enter/ modify/ delete) the Production Rules (via GUI).	PlPlanningUserUI; PlProductionRequest; PlDPR
S-PLS-00460	The PLANG CI shall maintain lists of Granules needed to satisfy Production Requests.	PlProductionRequest; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlDataGranule;
S-PLS-00470	The PLANG CI shall maintain information on the following: a. current processing status of all Production Requests received, b. current processing status of all Data Processing Requests generated, c. detected processing fault data.	PlProductionRequest; PlDataTypeCatalogue; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlDataGranule; MsEvent

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-00475	The PLANG CI shall maintain information on all Candidate and Active Plans generated.	PlPlanningWorkbenchUI; PlPlan; PlGroundEvents; PlGroundEvent; PlGroundActivity; PlDPRs; PLDPR; PlPGEActivity; PlResourceManager; PlActivePlan;
S-PLS-00490	The PLANG CI shall maintain Planning system fault data using fault isolation tools provided by the LSM.	MsEvent
S-PLS-00670	The PLANG CI shall provide (to the operations staff) the capability to enter, via GUI, "plan creation requests" that initiate creation of Candidate Plans.	PlPlanningWorkbenchUI; PlPlan;
S-PLS-00680	The PLANG CI shall provide the capability to generate multiple Candidate Plans.	PlPlanningWorkbenchUI; PlPlan;
S-PLS-00690	The PLANG CI shall create a Candidate Plan specifying a timeline for PGE execution that will satisfy Production Requests for Standard Products.	PlPlanningWorkbenchUI; PlPlan; PlGroundEvents; PlGroundEvent; PlGroundActivity; PlDPRs; PLDPR; PlPGEActivity; PlProductionRequest;
S-PLS-00710	The PLANG CI shall create a Candidate Plan based on the following: 1. Outstanding production requests, their priorities and estimated runtimes, 2. Ground events, their priority and estimated duration, 3. Planning production rules, 4. Mutual PGE accessibility of shared data, 5. Completion notification status messages from Data Processing.	PlPlanningWorkbenchUI; PlPlan; PlProductionRequest; PlGroundEvents; PlGroundEvent; PlGroundActivity; PlDPRs; PLDPR; PlPGEActivity; PlResourceManager;
S-PLS-00730	The PLANG CI shall have the capability to plan algorithm and calibration coefficient test time in the test environment.	PlPlanningWorkbenchUI; PlPlan; PlGroundEvent;
S-PLS-00740	The PLANG CI shall have the capability to schedule algorithm test Data Processing Requests that do not interfere with the operational production environment.	PlPlanningWorkbenchUI; PlPlan; PlResourceRequirement; PlDPRs; PLDPR; PlPGEActivity; PlResourceManager;
S-PLS-00760	The PLANG CI shall send electronic copies of the Candidate Plans and corresponding metadata to the designated local Data Server for storage and distribution.	PlPlanningWorkbenchUI; PlPlan; PlPublishedPlan; PlPlanMetadataFile; PlPlanASCIIReportFile; PlPlanBinaryReportFile;
S-PLS-00770	The PLANG CI shall provide (to the operations staff) the capability to enter, via GUI, a "Plan cancellation" request, indicating cancellation of the currently Active Plan.	PlPlanningWorkbenchUI; PlActivePlan; PlPGEActivities; PlPGEActivity; PlDPR; DpPrScheduler;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-00780	The PLANG CI shall generate Data Processing Request cancellations against previously submitted Data Processing Requests (if so directed by the operations staff), or upon activation of a new plan that no longer requires those requests.	PlPlanningWorkbenchUI; PlPlan; PlPGEActivity; PLDPR; DpPrScheduler;
S-PLS-00790	The PLANG CI shall send a Data Processing Request cancellation to the instance of the PRONG CI that received the original Data Processing Request.	PlPlanningWorkbenchUI; PlPlan; PlDPRs; PLDPR;
S-PLS-00800	The PLANG CI shall provide to the operations staff the capability to enter, via GUI, a "plan activation request" that identifies which Candidate Plan is to be activated.	PlPlanningWorkbenchUI; PlPlan; PlDPRs; PLDPR; PlPGEActivity; PlResourceManager;
S-PLS-00825	The PLANG CI shall have the capability to identify all available input data (as specified in the Active Plan) that is currently awaiting quality assurance information.	See Note 1
S-PLS-00827	The PLANG CI shall update the quality assurance status of input data (if applicable) to reflect an expired QA timeout period if its quality assurance information has not been received within specified time periods.	See Note 1
S-PLS-00830	The PLANG CI shall send Data Processing Requests (specified in an Active Plan) to a processing resource that can perform the processing, if the following applies: 1 All required input data (including metadata) is available,	PlSubscriptionManager; PlDPR; PlDataGranule; DpJobScheduler;
S-PLS-00840	The PLANG CI shall send electronic copies of the Active Plan and corresponding metadata to the designated local Data Server for storage and distribution.	PlPlanningWorkbenchUI; PlPlan; PlPublishedPlan; PlPlanMetadataFile; PlPlanASCIIReportFile; PlPlanBinaryReportFile;
S-PLS-00870	The operations staff shall manually submit Data Subscriptions for PGE input data to the appropriate Data Servers.	PlSubscriptionSubmitIF; PlDataTypes; PlDataType; IoAdAdvertisingSrv_C; IoAdServiceCollection_C; IoAdServiceAdvertisement_C; DsClSubscription
S-PLS-00872	The operations staff shall manually submit Data Subscriptions for L0 data to the Ingest Subsystem.	PlSubscriptionSubmitIF; PlDataTypes; PlDataType; AdCollection; Advertisement; DsClSubscription
S-PLS-00875	The PLANG CI shall receive Subscription Notices indicating availability of subscribed data.	EcMsgPsngCtrl; PlDataTypes; PlDataGranules; PlDataGranule; GlUR; DsClESDTReference;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-00880	The operations staff shall manually cancel Data Subscriptions for input data to PGEs that are no longer used, once they determine that the input data is not required by any other PGE.	PlSubscriptionSubmitIF; PlDataTypes; PlDataType; AdCollection; Advertisement; DsClSubscription;
S-PLS-01000	The PLANG CI shall receive a Data Processing Request Response message, acknowledging acceptance of the Data Processing Request forwarded to the PRONG CI.	PlDPR; PlActivities; DpJobScheduler;
S-PLS-01010	The PLANG CI shall receive "Complete Notification" status messages, indicating the completion status of Data Processing Requests.	PlDPR; PlActivities; DpJobScheduler;
S-PLS-01020	The PLANG CI shall receive responses to Data Processing Request cancellations indicating the completion status of the cancellation requests.	PlDPR; PlActivities; DpJobScheduler;
S-PLS-01030	The PLANG CI shall update the Active Plan with the current processing status of each Data Processing Request listed.	PlPlanningWorkbenchUI; PlPlan; PlPGEActivities; PlDPR; PlActivities; DpJobScheduler;
S-PLS-01040	The PLANG CI shall send the current processing status of Production Requests (for On-Demand Data Products) to the originating Data Server.	See Note 1
S-PLS-01200	The PLANG CI shall provide the operations staff with the capability to perform the following on-line functions, via GUI: a. Entry of product requests for standard products, b. Query / update / cancellation of production requests for standard products, c. Query status of production requests, d. Query / update of production rules and PGE information, e. Entry of plan creation requests, f. Entry of plan activation requests, g. Entry of plan cancellation requests, h. Query candidate / active plans and corresponding status, i. Entry of requests for processing log reports / production and data processing request status reports / resource utilization reports / planning workload status reports / management reports, j. Entry of ground events, k. Query / update of ground events.	PlPlanningWorkbenchUI;
S-PLS-01220	The PLANG CI shall have the capability to accept a request from the operations staff for scheduling algorithm and calibration coefficient test time in the test environment.	PlPlanningWorkbenchUI; PlPlan; PlGroudEvent;
S-PLS-01240	The PLANG CI shall support the display (via GUI) of Planning hardware and software detected faults to the operations staff.	MSS COTS

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-01245	The PLANG CI shall provide capability to make available (for review by all affected instrument teams) information related to product generation delays and production faults.	PlPlanningWorkbenchUI; PlPlan; PlPublishedPlan; PlPlanMetadataFile; PlPlanASCIIReportFile; PlPlanBinaryReportFile;
S-PLS-01250	The PLANG CI shall record detected hardware and software errors in a Planning processing log.	MsEvent
S-PLS-01260	The PLANG CI shall support the capability to generate Planning processing log reports (periodically and on request) for a specified time period.	DPS COTS
S-PLS-01270	The PLANG CI shall support the generation of Data Processing Request Status reports (upon request) that will provide Data Processing Request information based on the report generation parameters and the time period specified.	PlPlanningWorkbenchUI; PlPlan; PlDPRs; PLDPR;
S-PLS-01280	The PLANG CI shall support the generation of Production Request Status reports (upon request) that will provide Production Request information based on the report generation parameters and the time period specified.	PlPlanningUserUI; PlProductionRequest; PlDataTypeCatalogue; PlDataType; PlPGE; PlDPR; PlDataTypeRqt; PlInputDataType; PlDataGranule;
S-PLS-01290	The PLANG CI shall support the generation of resource utilization reports (periodically and on request).	MSS COTS
S-PLS-01300	The PLANG CI shall support the capability to generate PLANG CI processing workload and processing turnaround time reports (periodically and on request).	MSS COTS
S-PLS-01320	The PLANG CI shall make all reports generated available for review.	MSS COTS
S-PLS-01330	The PLANG CI shall restrict the functions available to operators depending on operations role, (e.g., to permit the resource manager role to enter ground events, but to restrict that role from being able to enter production requests).	PlApplication; PlPlResourceManagersUI; PlPlanningWorkbenchUI; PlProducitonRequestUI; PlUser;
S-PLS-01400	The PLANG CI shall accept the fault isolation tools for the PLANG CI.	PLANG CI;
S-PLS-01410	The PLANG CI shall forward faults detected in the Planning system to MSS.	PLANG CI;
S-PLS-01430	The PLANG CI shall send to MSS product scheduling, processing status and data quality information.	PLANG CI;
S-PLS-01440	The PLANG CI shall collect Fault Management Data and provide it to the MSS.	PLANG CI;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-01450	The PLANG CI shall collect Configuration Management Data and provide it to the MSS.	PLANG CI;
S-PLS-01470	The PLANG CI shall collect Accountability Management Data and provide it to the MSS.	PLANG CI;
S-PLS-01480	The PLANG CI shall collect Performance Management Data and provide it to the MSS.	PLANG CI;
S-PLS-01490	The PLANG CI shall collect Security Management Data and provide it to the MSS.	PLANG CI;
S-PLS-01500	The PLANG CI shall collect Scheduling Management Data and provide it to the MSS.	PLANG CI;
S-PLS-01600	The PLANG CI design and implementation shall have the flexibility to accommodate Planning expansion up to a factor of 3 in its capacity with no changes to its design, and up to a factor of 10 without major changes to its design. Such expansion in capacity or capability shall be transparent to existing algorithms or product specifications.	PLANG CI; PLNHW CI
S-PLS-01610	The PLANG CI shall be developed with configuration controlled APIs that will be capable of supporting development and integration of new algorithms developed at DAAC to support DAAC value-added production.	PLANG CI;
S-PLS-60010	The PLNHW CI shall support the hardware resource requirements of the PLANG CI and its interface requirements with the operations staff performing planning functions.	PLNHW CI;
S-PLS-60150	The PLNHW CI shall have provision for Initialization, Recovery, and an orderly shutdown.	PLNHW CI;
S-PLS-60160	Startup and initialization of the PLNHW CI shall be completed within 30 minutes (TBR).	PLNHW CI;
S-PLS-60170	Shutdown of the PLNHW CI shall be completed within 30 minutes (TBR).	PLNHW CI;
S-PLS-60180	The PLNHW CI shall have provision for a fault detection/ fault isolation capability without interfering with operations.	PLNHW CI;
S-PLS-60190	The PLNHW CI shall have a status monitoring capability.	PLNHW CI;
S-PLS-60320	The PLNHW CI shall support TBD transactions per day, as specified for each release and corresponding DAAC sites (TBR).	PLNHW CI;
S-PLS-60330	The PLNHW CI shall provide local storage of TBD GB (TBR).	PLNHW CI;
S-PLS-60340	The PLNHW CI shall provide a DBMS storage of TBD GB (TBR).	PLNHW CI;
S-PLS-60410	The PLNHW CI shall be capable of operating in a 24 hour per day, 7 days a week mode.	PLNHW CI;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-60420	PLNHW CI functions shall have an operational availability of TBD as a minimum and Mean Down Time (MDT) of TBD hours .	PLNHW CI;
S-PLS-60450	The PLNHW CI shall be capable of supporting system maintenance without impact to normal operations.	PLNHW CI;
S-PLS-60610	The PLNHW CI shall have provision for interfacing with one or more Local Area Networks (LANs).	PLNHW CI;
S-PLS-60620	The PLNHW CI shall have provision for interfacing with the Data Server subsystem.	PLNHW CI;
S-PLS-60625	The PLNHW CI shall have provision for interfacing with the Processing subsystem.	PLNHW CI;
S-PLS-60630	The PLNHW CI shall provide maintenance interfaces to support the function of System Maintenance.	PLNHW CI;
S-PLS-60640	The PLNHW CI shall provide operations interfaces to support the function of System Maintenance.	PLNHW CI;
S-PLS-60810	The electrical power requirements for PLNHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-60840	The air conditioning requirements for PLNHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-60850	The grounding requirements for PLNHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-60860	The fire alarm requirements for PLNHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-60870	The acoustical requirements for PLNHW CI equipment shall be in accordance with ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-60880	The physical interface requirements between PLNHW CI equipment and the facility shall be in accordance with ECS Facilities Plan (DID 302/ DV2).	PLNHW CI;
S-PLS-60890	The footprint size and the physical layout of PLNHW CI equipment shall be in accordance with the ECS Facilities Plan (DID 302/DV2).	PLNHW CI;
S-PLS-61010	The PLNHW CI shall support test activities throughout the development phase.	PLNHW CI;
S-PLS-61020	The following testing shall be performed on the PLNHW CI:	PLNHW CI;
S-PLS-61040	Internal testing shall be performed on the PLNHW CI which includes tests of hardware functions, and integration testing with other SDPS subsystems.	PLNHW CI;

Table A-1. Requirements Trace

L4 Rqmt ID	L4 Requirement Text	Object Class, CSC, or CI
S-PLS-61050	Internal testing shall be performed on the PLNHW CI to verify the internal interfaces to the Data Server, and Ingest subsystems.	PLNHW CI;
S-PLS-61080	The PLNHW CI shall be capable of supporting end-to-end test and verification activities of the EOS program including during the pre-launch, spacecraft verification, and instrument verification phases.	PLNHW CI;
S-PLS-61150	The PLNHW CI shall be capable of being monitored during testing.	PLNHW CI;
S-PLS-61210	The operating system for each Unix platform in the PLNHW CI shall conform to the POSIX.2 standard.	PLNHW CI;
S-PLS-61220	Each PLNHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm.	PLNHW CI;
S-PLS-61230	Each PLNHW CI POSIX.2 compliant platform shall have the following POSIX.2 user Portability Utilities installed at a minimum: man, vi.	PLNHW CI;
S-PLS-61240	Each PLNHW CI platform shall have the following POSIX.2 Software Development utilities installed: make, imake.	PLNHW CI;
S-PLS-61260	Each PLNHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell.	PLNHW CI;
S-PLS-61350	Each PLNHW CI POSIX.2 compliant platform shall have a screen capture utility.	PLNHW CI;
S-PLS-61530	The PLNHW CI shall contain the processing, storage, and interface resources to support the planning functions for the TRMM mission instruments of CERES and LIS.	PLNHW CI;
S-PLS-61610	Each PLNHW CI workstation platform shall provide a hard media device with a capacity of TBD GB for software and system maintenance and upgrade support.	PLNHW CI;

Note 1: The release allocation of this L4 requirement is the subject of a CCR that was pending at the time of publication of this document.

This page intentionally left blank.

Abbreviations and Acronyms

ADSRV	advertising service CSCI
AHWGP	Ad Hoc Working Group on Production
AI&T	algorithm integration and test
AITTL	algorithm integration and test tools (CSCI)
AM-1	EOS AM Project (morning spacecraft series)
ASCII	American Standard Code for Information Exchange Interchange
CASE	computer aided software engineering
CDR	Critical Design Review
CERES	Clouds and Earth's Radiant Energy System
CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSC	computer software component
CSCI	computer science configuration item
CSMS	Communications and Systems Management Segment (ECS)
DAAC	distributed active archive center
DBMS	database management system
DDSRV	document data server
DEV	developed code
DPR	data processing request
ECS	EOSDIS Core System
EDF	ECS development facility
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
GB	gigabyte (10^9)
GC	global change
GUI	graphic user interface
HCL	Hughes class library
HTML	Hyper-Text Markup Language
HWCI	hardware configuration item

I/O	input/output
L0	Level 0
LAN	local area network
LaRC	Langley Research Center (DAAC)
MB	megabyte (10^6)
MSFC	Marshall Space Flight Center (DAAC)
MSS	Management Subsystem
NOAA	National Oceanic and Atmospheric Administration
OO	object oriented
PDPS	planning and data processing system
PDR	Preliminary Design Review
PGE	Product Generation Executive
PLANG	production planning CSCI
PR	production request
RAID	redundant array of inexpensive disks
RDBMS	relational database management system
RMA	reliability, maintainability, availability
SCSI	Small Computer System Interface
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
U/I	user interface
UNIX	POSIX operating system
WAIS	Wide Area Information Server
WWW	World Wide Web